FINDING THE MINIMUM TEST SET WITH THE OPTIMUM NUMBER OF INTERNAL PROBE POINTS

BY

KWAN WAI WING, ERIC

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Philosophy in Electronic Engineering in the Graduate School of The Chinese University of Hong Kong

June 1996

Abstract

Observability can be increased by accessing internal nodes with the increasing use of E-Beam Testing. Conventional test generation techniques need to be modified if internal probe points are being considered. By critical path tracing, a Fault Dictionary can be generated to hold all the information of stuck-at-faults detected by the primary outputs or internal probe points. A network flow algorithm called the out-of-kilter algorithm which finds the minimum test set and optimum internal probe points has been implemented. In variation of this algorithm can find the minimum test set with a fixed number of internal probe points. In order to obtain the minimum test set with the optimum number of internal probe points, an analysis of experimental results generated from this algorithm shows a way to improve an existing algorithm. Practical approach will also be suggested.

Acknowledgment

I would like to express my deepest gratitude to my supervisor, Dr. Oliver C.S. Choy for his supervision and useful comments over the whole research. His guidance, advise and support are really helpful throughout my study of the course at the Chinese University of Hong Kong.

In particular, special thanks to my family for their patience and understanding.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Page

1. Introduction

1.1 Background

The main purpose of test generation is to produce a number of test vectors which distinguish good chips from bad ones. The test generation process becomes difficult as the increasing integration and complexity of very large scale integrated (VLSI) circuits, it costs more and take longer to complete the generation process. Actually, the source of difficulties are the poor controllability and observability of VLSI circuits. The conventional method bases on the controlling of logic state at the primary input and observing the logic state at the primary output.

The crosscheck design technology [1,2] and E-beam tester [3,4,5] improve the observability significantly by providing the ability to access internal nodes. The crosscheck design technology has an embedded test structure which can access internal test points of VLSI circuits by memory-array like addressing scheme. The E-beam tester uses an electron beam prober to observe the logical value of an internal signal line which is running at the top metal level of the circuit.

The main advantage of high observability testing is to reduce the number of test vectors. Conventional test generation techniques are based on external testing and commercial software tools which do not support internal probing. The problem for high observability testing is to find an appropriate testing generation method which generates test vectors as well as internal probe points.

1.2 E-beam testing and test generation algorithm

There are various types of high observability testing and we will concentrate on the test generation for E-beam testing, since it is a non-contact testing which introduces an infinite impedance with no capacitance and non loading of the circuit. It also allows both direct and random access to the internal nodes. The flexibility of choosing probe point is highly appreciated by the new strategy of testing.

In the conventional test generation algorithm, no internal probe point is put into consideration and the observability relies on a fixed number of output pins. With the introduction of high observability testing [6], many internal test points are available, and their number and locations per test vector are arbitrary.

There is an existing E-beam testing algorithm [7,8] which was developed to generate test vectors with corresponding test points. The algorithm makes sensible selection of test points for each test vectors. It starts with a circuit description file and a fault list consisting of all stuck-at faults. A seed line is then chosen and critical paths are in turn determined by successive steps of propagation, justification and potential probe point addition. Two fundamental steps to create critical path backward to low level lines, and, second, to create a critical path forward to primary output. Justification(l) is used to create critical lines which are one level below line. Propagation(l) is employed to create lines which are the outputs of the gate with line l as its input. After the removal of redundant probe points, a test vector with probe points are generated. This procedure is repeated until all stuck-at faults are detected.

1.3 Motivation of this research

It is not surprising to find a VLSI design without a satisfactory test due to the excessive testing cost. Researchers are working on both hardware and software solutions to modify their existing tools in order not to be left behind by the fast increasing integration of VLSI.

Conventional automatic test pattern generation [9] is based on the responses observed at the output pins. With the introduction of high observability testing, there exists a need for development in the area of test pattern generation. The new software should put the addition of internal probe points into consideration to reduce the number of test vectors and hence the testing cost.

With the existing software tools in which internal probe points are being considered, it is well proven that the number of test vectors can be reduced. However, to what extend can these test vectors be reduced into the lower bound or the minimum of this test generation becomes the area we are interested in. Correspondingly, the number of test vectors generated by conventional method are set at the upper bound in this study. Having the advantage of getting to know the upper bound and the lower bound of the solution, a lot of insights can be gained by studying at the gray area in between.

In this research, a network flow algorithm is implemented to find the minimum test set and optimum number of internal probe points by extracting the information from a Fault Dictionary. An optimum solution can also be obtained with a fixed number of internal probe points. With those results, further recommendation and implementation can be suggested to improve the existing test generations.

1.4 Out-of-kilter Algorithm

It is well known that network flow programming [10,11,12] is an efficient method for industrial engineering analysis. A broad class of problems can be conveniently classified under the category of constrained network flows. One method of solving this class of problems is through the use of the out-of-kilter algorithm, OKA [13]. To apply the algorithm, a network must be defined that will yield an optimum solution. A closed loop network flow and a conservation of flow with initial value are necessary for proper representation of the problem.

The modeling tool for the out-of-kilter arc is described by three numbers : cost per unit flow, lower bound to flow and upper bound to flow. In this thesis, we formulate the circuit into a network flow model by using the cost of the flow through an arc to describe the characteristic of the circuit. We make use of the out-of-kilter algorithm to minimize the cost of the flow by selecting (upper bound) or not selecting (lower bound) to evaluate the optimum number of test vectors for testing a VLSI circuit.

1.5 Outline of the remaining chapter

A high observability testing method, which is called the E-Beam testing, is introduced in the next chapter. Chapter 3 describes the procedure to generate the Fault Dictionary. An exhaustive method is introduced by generating every possible test vectors and put into the fault dictionary.

Chapter 4 describes the mathematical model of a network flow modeling and the out-of-kilter algorithm.

Chapter 5 describes the method to turn the Fault Dictionary into a network flow model and hence find the solution of the minimum test set. The application of the algorithm to some simple circuits is included in Chapter 6.

A more realistic approach will be discussed in Chapter 7 to significantly reduce the size of the Fault Dictionary to speed up computation time. Improvement to the existing test pattern generation algorithm is also discussed in this Chapter.

Conclusion and suggestions for further research are discussed in Chapter 8.

2. Electron Beam testing

2.1 Background and Theory

The foundation of electron beam testing (EBT) were laid in the 1950s. All the studying activities follow the first demonstration of a useful scanning electron microscope (SEM) by D. McMullan who has proved sufficiently promising for further investigation or development of the SEM. It leads to the gained acceptance as a standard technique for integrated circuit testing and substitute the mechanical probing which is unfeasible to handle a very small device.

After the foundation created at Cambridge University in the 1950s. The 1960s has led to a major contribution to modern high technology. Through the 1970s to 1990s, the use of scanning electron microscopes and similar instruments from the semiconductor industry became widespread around the world and in all respects of research, development, testing, quality control and even in publicity for a company's products.

The basic principle of EBT is to observe the phenomenon of voltage contrast in which a potential applied to a specimen in the SEM causes the intensity of its image to change. Figure 2.1 shows a conductor being probed at 0 volt (Logic zero).

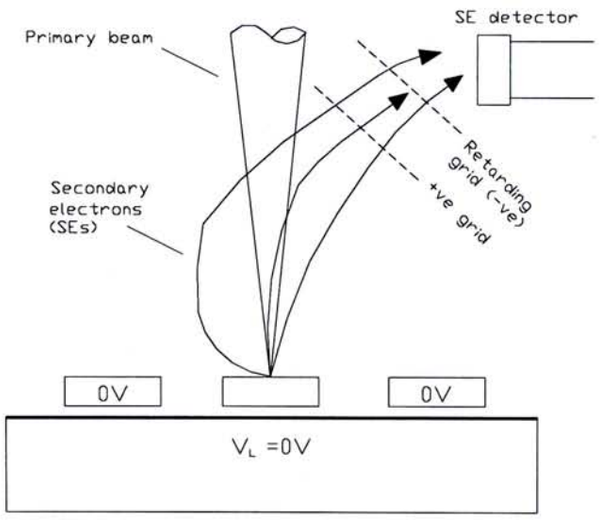


Figure 2.1 Conductor being probed biased at 0 Volt

A large portion of the higher energy secondary electrons with sufficient energy is passing through the retarding grid. They are detected by the SE detector and Figure 2.2 shows the distribution of the secondary electrons being detected.

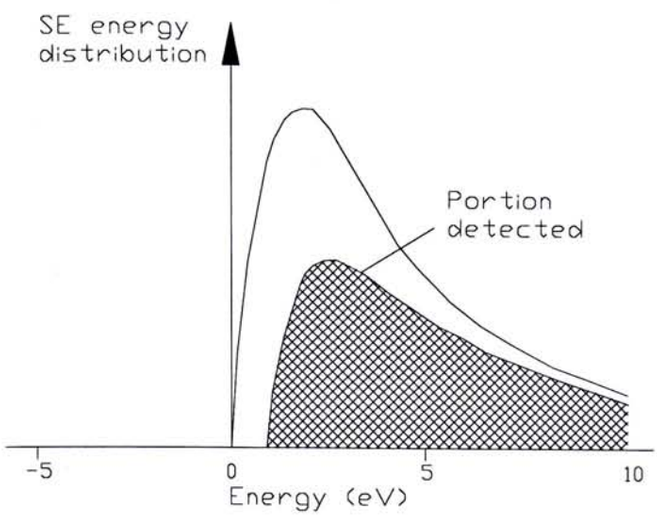


Figure 2.2 Large Portion of Secondary Electron are detected

For the conductor being probed at 5 Volt (Logic one) as shown in Figure 2.3:

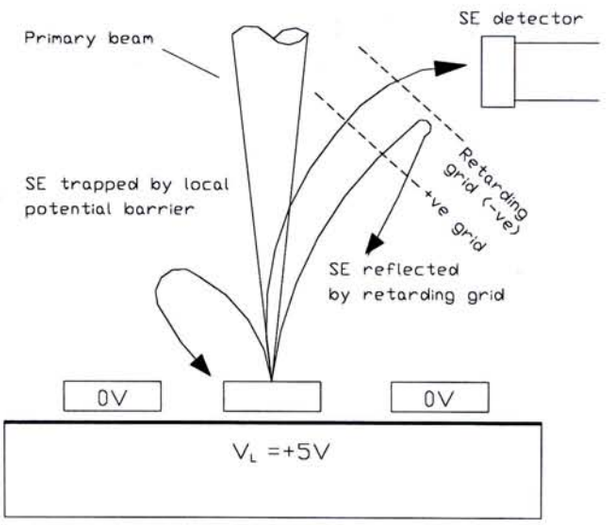


Figure 2.3 Conductor being probed biased at 5 Volt

since the lower energy secondary electrons are trapped by local barrier, the detected portion of the spectrum is smaller as shown in Figure 2.4:

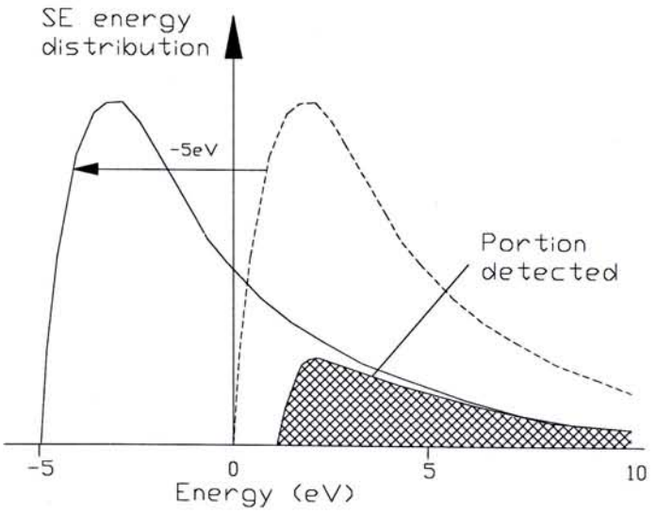


Figure 2.4 A smaller detected portion of the spectrum

Thus, scanning the beam over an IC to obtain a secondary electron image, logic one appears dark while a logic zero appears bright. This forms a powerful test methodology based on the contrast of an image.

2.2 Principles and Instrumentation

Since EBT is built on the foundation of scanning electron microscopy, it shares many common principle and element of instrumentation. Figure 2.5 shows an electron optical column, which is the major components of an EBT.

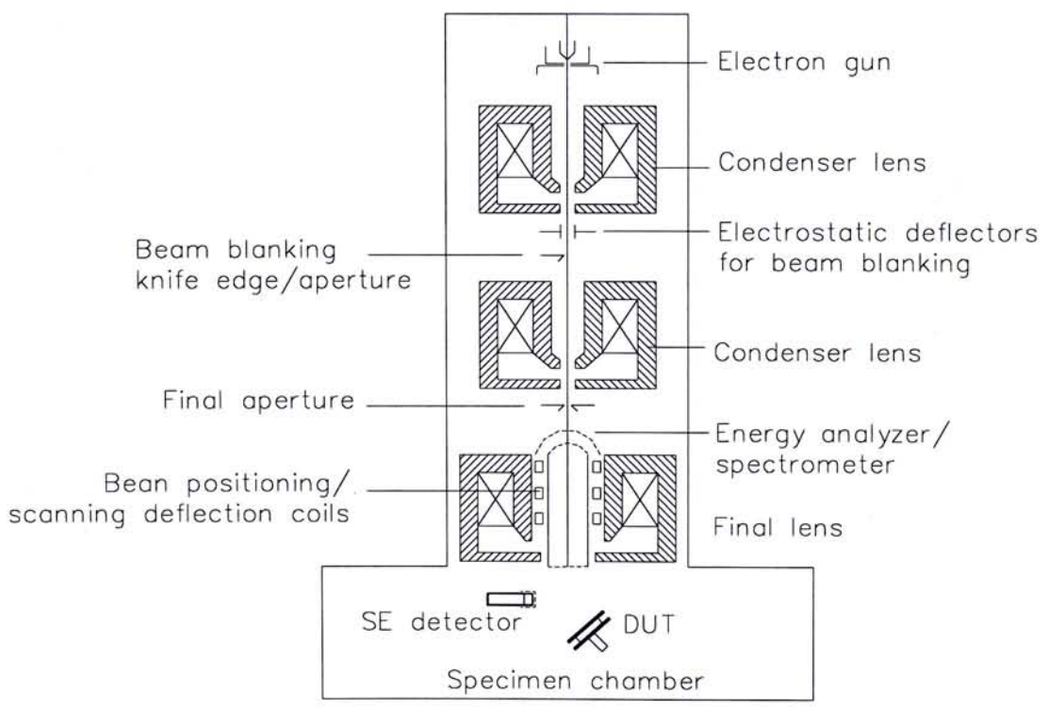


Figure 2.5 Major components of an EBT electron-optical column

The component parts of an electron beam tester column [14] consists of an electron source, a beam blanker, condenser lens, an objective aperture, a beam scanning deflection coils, a specimen chamber and a secondary electron detector.

An electron source with low beam energy ranging from 0.5 - 2.5 KeV is accelerated while a beam blanker stops the electron beam between two inspection locations. The condenser lens focus the electron beam onto the specimen as a small spot. An image can be observed when the electron beam is scanning across the specimen and the logic level will be determined by the voltage contrast phenomenon. The function of an objective aperture is to reshape an elongated spot into a circular one to reduces the lens aberration effects. The specimen chamber contains an adjustable specimen stage and an electron detector counts the number of secondary electrons being detected.

2.3 Implication of internal IC testing

EBT offers a number of desirable features that have led to its development as a commercial tool for internal IC testing [15,16]. It is not only possible to locate the area of interest by imaging the IC with a scanning beam, but also to position the electron probe accurately and rapidly at a measurement node [17,18].

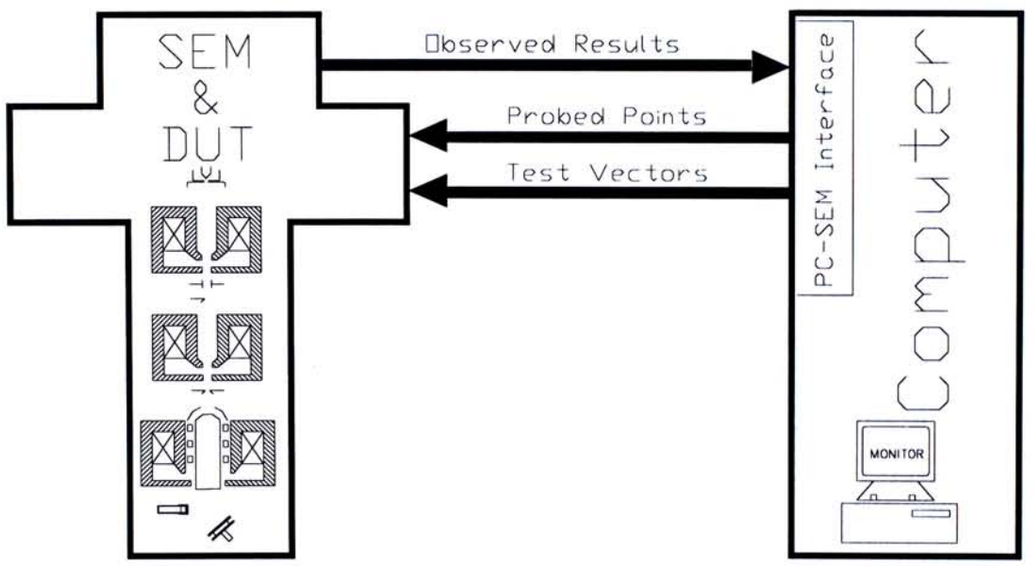


Figure 2.6 Configuration of an EBT

The main elements of an E-beam tester system are shown schematically in Figure 2.6. It consists of a computer and a PC-SEM interface [19,20] which acts as a bridge between the SEM and PC to give out SEM control signals and receive data. A digitized gray level signal of the probe location will be transmitted back to the computer for analysis.

2.4 Advantage of Electron Beam Testing

The ability of Electron Beam testing to observe the internal functioning of a device was recognized as a valuable asset as IC's are becoming increasingly complex. It's advantages include the non-destructive and non-capacitive loading of the testing specimen. The integrated circuit can be tested under normal working conditions. The voltage contrast phenomenon determines the logic value of the line while E-beam is directed to the point of interest and secondary electron emitted are being collected. Since the testing procedure are computerized, the controlling of the SEM and voltage inspection of the internal points are all processed by the computer.

As the observability of the circuit and the fault coverage of each test vector increase, the testing cost will be reduced. It reduces the cost of test pattern generation and the cost of the time consumed during the test. With the above advantages, it confirms its level of usefulness in the IC diagnosis. Based on the assumption that internal lines can be determined, we continue our research for a better test pattern generation software which supports E-beam testing.

3. An exhaustive method to minimize test sets

There are various types of test pattern generation methods and they do not guarantee a true minimum. In order to arrive at the best test set, we need to analyze all the possible test vectors. In this Chapter, we describe how we put elements into the Fault Dictionary. If we put all possible test vectors and perform optimization to reach an optimum solution, it becomes an exhaustive method.

3.1 Basic Principles

3.1.1 Controllability and Observability

Controllability is a measure of how easy the logic of an internal circuit node can be controlled from the primary input. Observability is a measure of how easy the internal circuit node can be observed at the output pins. The idea to modify the design of a circuit to enhance its controllability and observability lead to a reduction in deterministic test generation costs.

With the introduction of logic value observation inside an internal signal lines [21], the value of observability can be increased significantly. It results in an increase of testability and a decrease of the number of test vectors.

Under high observability environment, the test generation needs to be modified. Hence, a good test algorithm is required.

3.1.2 Single Stuck at Fault Model

A fault model is often used to represent the physical defects in a digital circuit. The Single-Stuck at Fault Model is also referred to as the classical or standard fault model because it has been the first and the most widely used model. In the model, it is also assumed that only one type of fault is present at a time and permanent faults are being considered instead of intermittent and transient faults.

Generally, a short circuit between the power or ground will stuck the signal line remaining at a fixed logical state. A line l stuck at a logic value v (0 or 1) can be treated as cutting the line l and setting a constant signal v to the output of the line l. This line l is said to be stuck-at-v. Or, the line l has a stuck-at-v fault. Single stuck at fault model assumes only one line to be faulty at a time and it is the most popular fault model used in gate level simulation, test pattern generation and fault simulation.

In single stuck-at-fault model, faults are often restricted to the input or output lines of the logic gates. For an n-line circuit having all binary values, there are 2n distinct possible stuck-at-faults. It is also common in fault simulation and test generation procedures such as critical path tracing [22] and D-algorithm [23]. These methods compute the circuit to enable the fault to be sensed at an primary output.

The single stuck-at-fault model can detect many non-classical faults as well, it has high effectiveness in the testing of digital circuits. Tests for stuck-at-faults tend to exercise all logic gates of a circuit. For example, an n-input AND gate has 2(n+1) distinct stuck-at-faults associated with its input and output lines. A unique set of at least n+1 test patterns are sufficient to detect all the stuck-at-faults. Stuck-at-faults based tests tend to apply almost all possible input patterns to the gates. With these test patterns exercising each gate, most physical faults are likely detected if an incorrect logic signal appears at the gate output.

3.2 Fault Dictionary

3.2.1 Input Format

ISCAS'85 netlist format is a standard netlist format which is used by many researchers as a basis for fault simulation and test generation. In our test circuits, we follow the ISCAS'85 netlist format to describe the lines by node. One of the node is shown as follows.

address	name	type	fanout	fanin
1	1gat	inpt	1	0

where

address - a unique number that differentiates the node line from the others inside the circuit.

name - a string of characters which is a readable information about the node.

type - a function performed by the gate which drive the node.

fanout - a unique number which tells the quantity of the output pin of the gate.

fanin - a unique number which tells the quantity of the input pin of the gate.

For the node which is shown above, it carries the information of :

- the node address is 1
- the node name is 1gat
- the node type is inpt
- the number of fanout is 1
- the number of fanin is 0

Circuit description of a logic module is shown:

```
1     1gat   inpt  1     0
2     2gat   inpt  2     0
3     3fan   from    2gat
4     4fan   from    2gat
5     5gat   inpt  1     0
6     6gat   inpt  1     0
7     7gat   inpt  2     0
8     8fan   from    7gat
9     9fan   from    7gat
10    10gat  inpt  1     0
11    11gat  inpt  1     0
12    12gat  inpt  1     0
13    13gat  not   1     1
      3
14    14gat  not   1     1
      8
15    15gat  and   1     2
      1        13
16    16gat  and   1     2
      4        5
17    17gat  and   1     2
      6        14
18    18gat  and   1     2
      9        10
19    19gat  nor   2     2
      11       12
20    20fan  from    19gat
21    21fan  from    19gat
22    22gat  or    1     2
      15       16
23    23gat  or    1     2
      17       18
24    24gat  not   1     1
      20
25    25gat  and   1     2
      22       24
26    26gat  and   1     2
      21       23
27    27gat  or    0     2
      25       26
```

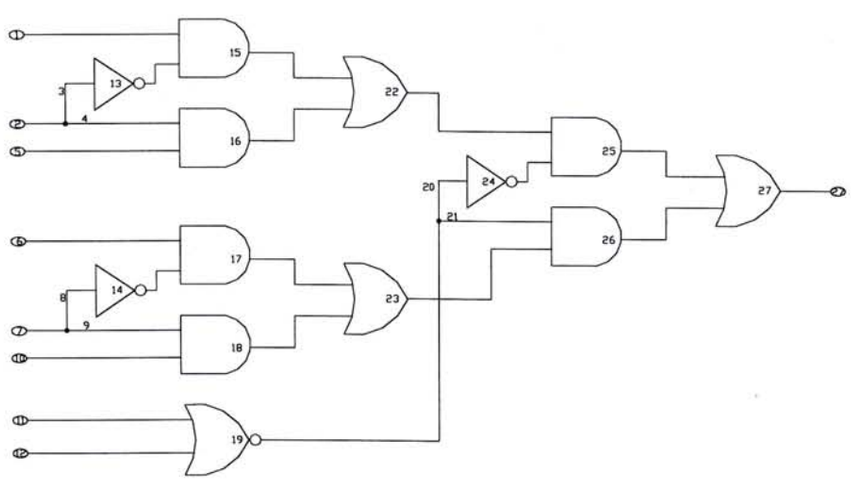


Figure 3.1 Circuit SC7

3.2.2 Critical Path Generation

Critical Path tracing method was originally developed as an alternative to fault simulation [22,24]. It is further developed to handle sequential circuit [25]. A critical path is similar to a sensitized path defined by the concept of critical values.

Definition: A line l has a critical value v in the test t iff t detects the fault l stuck-at-$\overline{v}$. A line with a critical value in t is said to be critical in t.
Definition: A gate input i is sensitive if the complement value of i changes the value of the gate output.

For every input vector, critical path tracing first simulates the fault free circuit, then it determines the detected faults by ascertaining which signals are critical. Consider the circuit "SC1.isc" in Figure 3.2,

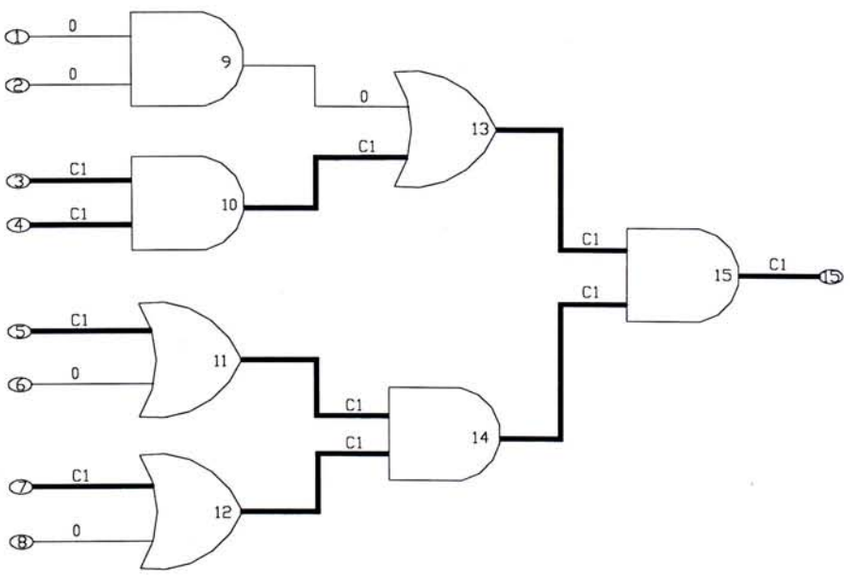


Figure 3.2 Example of critical paths

there are four critical paths: path(3,10,13,15), path (4,10,13,15), path(5,11,14,15) and path(7,12,14,15). These lines are all critical with logical value equal to one. Ten lines are critical at one with the input test vector pin(1,2,3,4,5,6,7,8) = vector(0,0,1,1,1,0,1,0). Therefore, it can detect stuck-at-0 fault of lines 3, 4, 5, 7, 10, 11, 12, 13, 14 & 15.

3.2.3 Probe point insertion

Consider the circuit shown in Figure 3.3, it has two critical paths (1,9,13,15) and (3,10,13,15) which can reach the primary output. It can detect stuck-at-1 fault of the lines 1,3,9,10,13 & 15 at the primary output. There are also critical paths (6,11) and (7,12) which stop at gate 14.

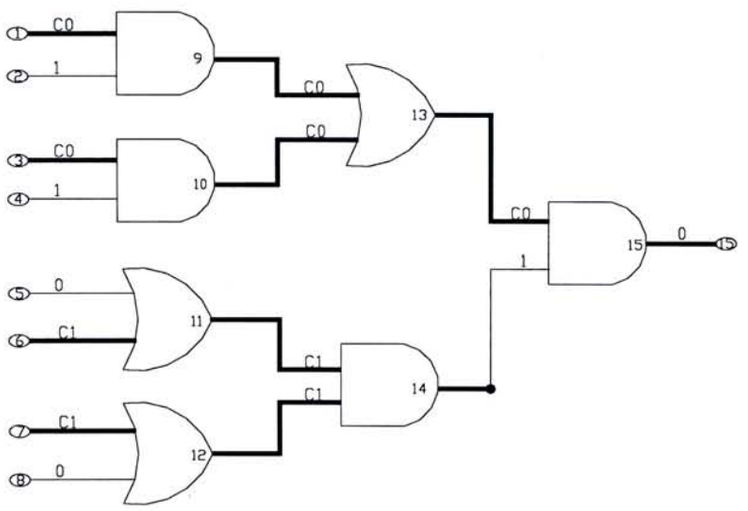


Figure 3.3 Example of critical path with probe point insertion

If we place a probe point at line 14, we can detect a stuck-at-0 fault for line 6,7,11 & 12. With the observation at primary output and an additional probe point at line 14, This test vector pin (1,2,3,4,5,6,7,8) = vector (0,1,0,1,0,1,1,0) can detect stuck-at-0 fault at line 6,7,11,12 and stuck-at-1 faults at line 1,3,9,10,13 & 15. For every test vector, we place a probe point wherever a critical path stops inside the circuit. This method ensures that we consider every possible way of probe point adding. And this vector ensures maximum coverage of stuck-at-faults occurring inside the circuit.

3.2.4 Formation of Fault Dictionary

Fault Dictionary is setting up to describe the characteristic of the circuit. All possible test vector are placed at the primary input and the procedure follows as mentioned in sections 3.2.2 and 3.2.3.
The execution of the algorithm steps are shown below and the flow chart is shown in Figure 3.4 :

Step 1 : Read the circuit description of the circuit SC7.
Step 2 : Get the first test vector with pin(1,2,3,4,5,6,7,8) = vector(0,0,0,0,0,0,0,0) to the primary input.
Step 3 : Execute fault-free simulation.
Step 4 : Critical path tracing from the primary input to the primary output and sensitive lines are marked.
Breakthrough : Mark '8' & '9' to represent stuck-at-1 and stuck-at-0 detected by primary output.
Non-Breakthrough : Add a probe point to the output of the non-breakthrough gate. Mark '4' & '5' to represent stuck-at-1 and stuck-at-0 detected by this probe point.
Step 5 : Get another test vector and back to step 2 until the last test vector is implemented.

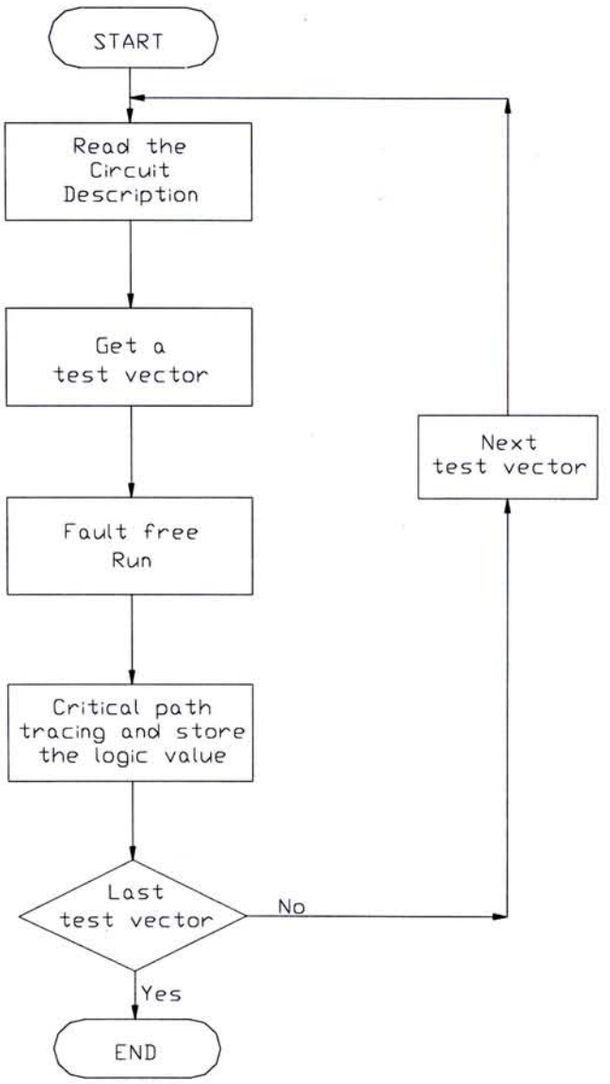


Figure 3.4 Fault Dictionary generation flow chart

The result is marked down as a format as

```
[              A                ] - [B] - [C]
.[               D                ] - [E] - [F]
```

where

A: Logic states of all the lines in a fault-free circuit.
B: Maximum number of probe points.
C: The test vector coded in decimal value.
D: Sensitivity of each line with respect to the probe point located by E.
E: Probe point location.
F: No. of lines made critical by respective probe point.
.: Identify the entry for a probe point.

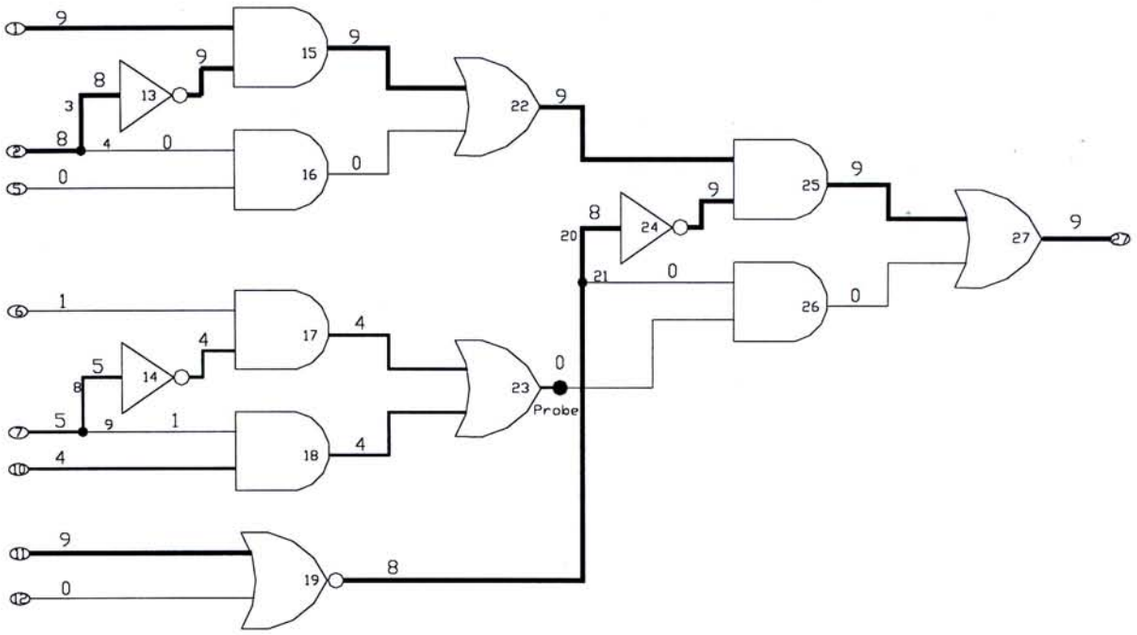


Figure 3.5 Circuit description of an element in the Fault Dictionary

One element of the Fault Dictionary for the circuit is shown:

```
9880015514909490448809099O9-1+89
.0000005504000400440000000O0-23+6
```

where

[0,1]	are the ordinary logic values.
[4,5]	are the critical 0,1 value detected at an added probe point.
[8,9]	are the critical 0,1 value detected at the primary outputs.
[-1]	show that there is at most one probe point for this vector.
[.line]	the critical path detected by adding a probe point at line 23.

A Fault Dictionary of Circuit SC1 is shown in Appendix A and Circuit SC7 is shown in Appendix B. With the Fault Dictionary which holds all the test information of the circuit, a mathematical tool is necessary to find an optimum solution for a minimum test set with optimum number of internal probe points.

In the following Chapter, a mathematical tool called the Out-of-kilter algorithm is introduced to solve the minimum test set problem.

4. Mathematical Model - Out of kilter algorithm

4.1 Network Model

Network flow modeling [26,27] is one of the most useful techniques in industrial engineering analysis. A broad class of these problems such as inventory control, scheduling, allocation and production control can be conveniently classified under category of constrained network flows.

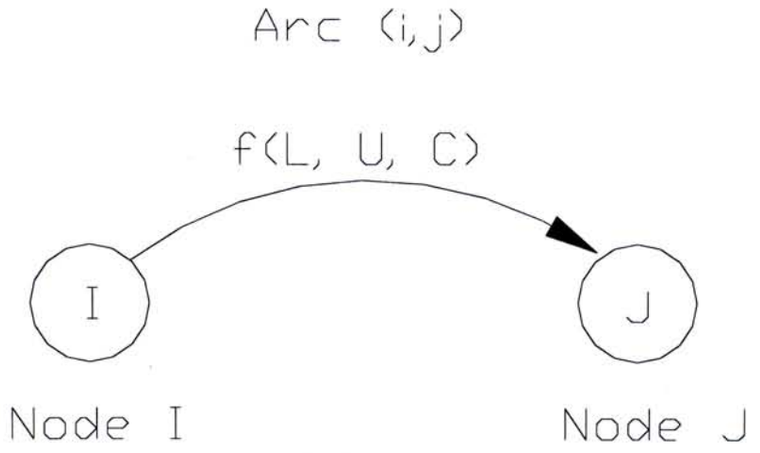


Figure 4.1 A basic entity of a network

A basic entity of a network diagram is shown in Figure 4.1 and it consists of nodes and arcs. A node is a termination point which generates and consumes flow. A node which generate flow is called a source node and a node which consume flow is called a sink node. An arc are the lines that connect the various nodes together in a network and sometimes called branches. A network is a set of connected arcs and nodes generally representing a physical process in which units move from source to sink. A typical network is shown in Figure 4.2 which consists of multiple sources and sinks.

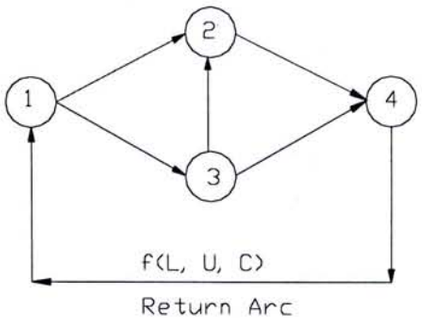


Figure 4.2 A typical network flow

A circulation is an assignment of flow to arcs such that flow is conserved at each node. That is, the total flow entering the node is equal to the total flow leaving the node. The out-of-kilter algorithm deals with circulation and it is often necessary to add an additional arc to connect the sink with the source. This additional arc is called return arc and the sink and source are called super sink and super source respectively.

A capacitated arc is characterized by a finite lower bound L, a finite upper bound U and an associated cost C per unit flow. The actual flow through each arc can be any flow between the upper and lower limits, as long as any additional constraints are not violated. Figure 4.1 shows a capacitated arc

where f_{ij} = flow through arc (i,j)
L_{ij} = lower capacity on arc (i,j)
U_{ij} = upper capacity on arc (i,j)
C_{ij} = cost associated with shipping one unit of flow from node i to node j

The out-of-kilter algorithm (OKA) is an iterative procedure to find the circulation in a given capacitative network which minimizes the total cost of all flows passing through the arcs of the network.

4.2 Linear programming model

For all nodes with external flows set to zero and conservation of flow required at each node, the network flow problem can be represented as a special linear programming problem [28]. Since the cost of shipping one unit across arc (i,j) is C_{ij}, the problem becomes one of the minimizing total cost.

$$\text{Min} \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}f_{ij}$$

subject to

Conservation of flow

$$\forall i = 1,2,.....,m, \quad \sum_{j=1}^{n} f_{ji} - \sum_{j=1}^{n} f_{ij} = 0$$

Lower bounds

$$\forall i = 1,2,.....,m,$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \geq L_{ij}$$

Upper bounds

$$\forall i = 1,2,.....,m,$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \leq U_{ij}$$

Non negative flow

$$\forall i = 1,2,.....,m,$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \geq 0$$

To form the dual of this linear programming problem, we assign Π_i to the conservation of flow constraint for node i, α_{ij} to the upper bound constraint for arc i to j and β_{ij} to the lower bound constraint to the arc i to j. The dual problem becomes:

$$\text{Min} \sum_{i=1}^{m}\sum_{j=1}^{n} Uij\alpha_{ij} - L_{ij}\beta_{ij}$$

subject to

$\Pi_i - \Pi_j + \alpha_i - \beta_j \geq - C_{ij}$ $\forall i,j \in N$
Π_i unrestricted $\forall i \in N$
$\alpha_{ij} \geq 0$ $\forall i,j \in N$
$\beta_{ij} \geq 0$ $\forall i,j \in N$ where N is the set of Nodes

Given a primal solution F and a partial dual solution Π, we find that the two solutions are optimal for their respective problem if the following conditions exist.

1. Primal feasibility

$P_1 : \sum_{j=1}^{n} f_{ji} - \sum_{j=1}^{n} f_{ij} = 0$ (Conservation of flow)

$P_2 : L_{ij} \leq f_{ij} \leq U_{ij}$ (Capacity constraints)

2. Dual feasibility

D_1: $\Pi_i - \Pi_j + \alpha_i - \beta_j \geq - C_{ij}$ for all $(i,j) \in S$
D_2: $\alpha_{ij} \geq 0$ for all $(i,j) \in S$
D_3: $\beta_{ij} \geq 0$ for all $(i,j) \in S$ where S is the set of arcs

3. Complementary Slackness

C_1: if $\Pi_i - \Pi_j + \alpha_i - \beta_j > - C_{ij}$, then $f_{ij} = 0$
C_2: if $\alpha_{ij} > 0$, then $f_{ij} = U_{ij}$
C_3: if $\beta_{ij} > 0$, then $f_{ij} = L_{ij}$

The OKA algorithm requires an initial flow solution that satisfies condition P_1 but not necessarily condition P_2. The initial node potentials are arbitrary so condition 3 is usually not satisfied. Condition 2 is not used in the algorithm but are useful for postoptimality sensitivity analysis. If a feasible solution exists, the ultimate attainment of an optimal solution in a finite number of iteration is guaranteed.

4.3 Kilter states

An equivalent formulation of the conditions for optimality is given by the following relationships:

I. If $\Pi_j - \Pi_i > C_{ij}$, then $\alpha_{ij} > 0$ and $f_{ij} = U_{ij}$

II. If $\Pi_j - \Pi_i < C_{ij}$, then $\beta_{ij} > 0$ and $f_{ij} = L_{ij}$

III If $\Pi_j - \Pi_i = C_{ij}$, then $U_{ij} \geq f_{ij} \geq L_{ij}$

provided that we choose

IV. $\alpha_{ij} = \max[0 \; ; \; \Pi_j - \Pi_i - C_{ij}]$

V. $\beta_{ij} = \max[0 \; ; \; -\Pi_j + \Pi_i + C_{ij}]$

and

VI. $\sum_{j=1}^{n} f_{ji} - \sum_{j=1}^{n} f_{ij} = 0$

In terms of seeking the optimum results through successive perturbations of the solution vector, these conditions are very efficient since only conditions I, II, III need be evaluated, and these do not involve the dual variables α and β of the relationships IV and V.

Assuming that conditions IV and V are satisfied, and using $C'_{ij} = C_{ij} + \Pi_i - \Pi_j$. Conditions I, II, III and VI can be put in a more convenient form:

K_1 : If $C'_{ij} < 0$, then $f_{ij} = U_{ij}$

K_2 : If $C'_{ij} > 0$, then $f_{ij} = L_{ij}$

K_3 : If $C'_{ij} = 0$, then $U_{ij} \geq f_{ij} \geq L_{ij}$

K_4 : conservation of flow is satisfied

If two nodes i and j, and their connecting arc, satisfy optimality condition K_1, K_2, or K_3, that arc is said to be in-kilter. If an arc does not satisfy either K_1, K_2, or K_3 that arc is said to be out-of kilter. There are totally nine kilter states. Three of them are in-kilter states and six of them are out-of-kilter states as shown in Table 8.1.

State	C'_{ij}	f_{ij}	In Kilter?	Why?
α	$C' > 0$	$f = L$	Yes	Satisfies K_2
β	$C' = 0$	$L \le f \le U$	Yes	Satisfies K_3
δ	$C' < 0$	$f = U$	Yes	Satisfies K_1
α_1	$C' > 0$	$f < L$	No	Violates P_2 & K_2
β_1	$C' = 0$	$f < L$	No	Violates P_2
δ_1	$C' < 0$	$f < U$	No	Violates K_1
α_2	$C' > 0$	$f > L$	No	Violates K_2
β_2	$C' = 0$	$f > U$	No	Violates P_2
δ_2	$C' < 0$	$f > U$	No	Violates P_2 & K_1

Table 4.1 Possible Kilter state of an arc

For each state, we can determine whether the conditions for optimality are satisfied. Those that do satisfy the conditions are called in-kilter states (α, β, δ). Those that do not are called the out-of-kilter states (α_1, β_1, δ_1, α_2, β_2, δ_2). An arc is identified as in-kilter or out-of -kilter on the basis of its kilter state. These kilter states are summarized in a pictorial fashion in Figure 4.3.

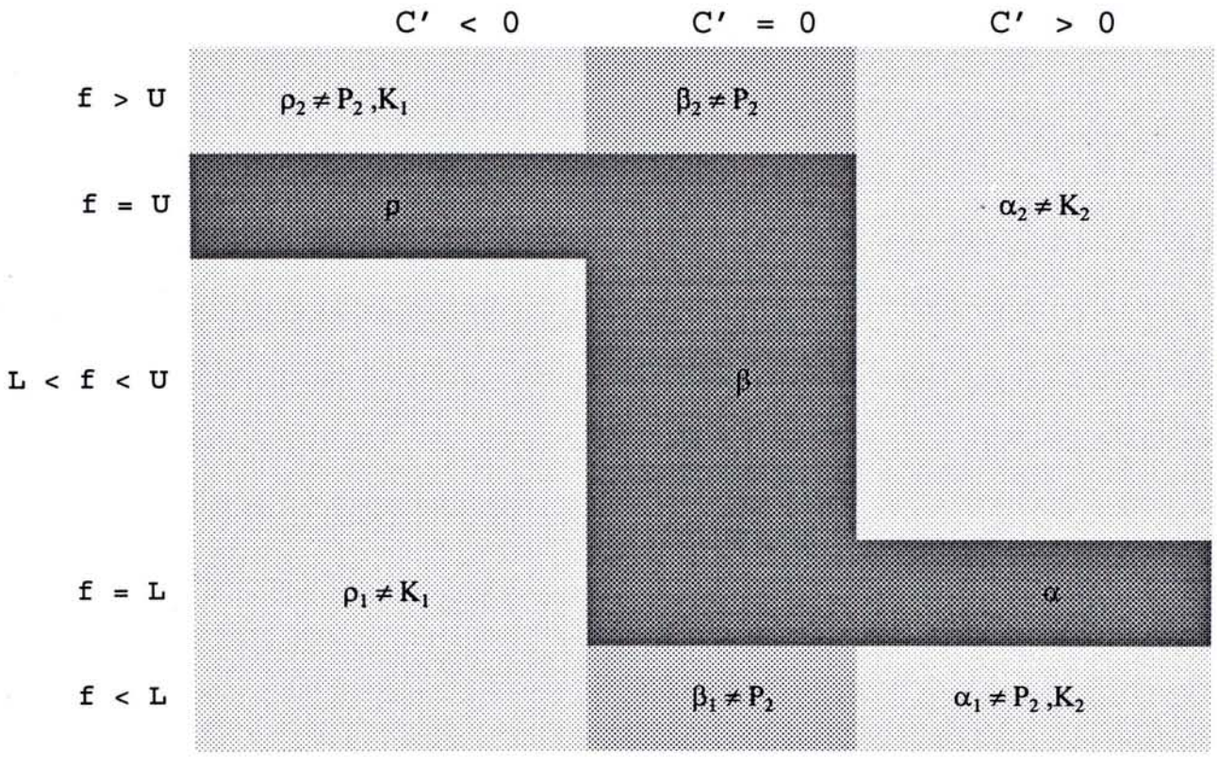


Figure 4.3 Pictorial representation of the Kilter states

4.4 Flow change

The OKA assumes that flow is conserved at each node. When the initial solution satisfies this condition, all subsequent solutions will also satisfy the condition since flow is always changed on a cycle. Investigation for a procedure is necessary to bring an arc from an out-of-kilter state to an in-kilter state without any in-kilter arc are thrown out-of-kilter or out-of-kilter arc are thrown further out-of-kilter. This procedure is called labeling procedure.

Consider an arbitrary arc, it will fall in one of the line mutually exclusive states as specified in Table 4.1. In order to bring an arc in-kilter, the arc is to be increased in either state α_1, β_1 or δ_1. For an arc to be decreased, the arc must be in either state α_2, β_2 or δ_2. If an arc is found to be in either state α, β or δ, that arc is in-kilter and its flow should not be altered. The exception is the state β, in which the flow might be increased or decreased without violating any conditions.

When an out-of-kilter arc is chosen, the flow across that arc is adjusted to bring the arc in-kilter. To preserve conservation of flow, an alternative path should be labeled by the above labeling procedure. If the alternative path is found, a breakthrough occurs. The flow is updated by following the labeling routes. If the alternative path is not found, a non-breakthrough occurs. All the flows will not be updated and another kilter arc will be selected.

4.5 Potential change

When a non-breakthrough occur and the algorithm fails to bring arc into an in-kilter state. Recall that the state of an arc is uniquely determined by checking $C'_{ij} = C_{ij} + \Pi_i - \Pi_j$, so that a change in the Π value affects the nine possible states of an arc. According to the procedure by which the dual problem was formally established, each and every node has a Π value associated with it, so that there are exactly n dual Π variables for a network-flow problem with n nodes. Consider two sets with A are the set of all labeled nodes and A′ are the set of all unlabeled nodes.

Case 1: Let B be the set of all arcs originating at a node in A terminating at a node in A′ with $C' > 0$ and flow less than or equal to the upper bound.
Case 2: Let B′ be the set of all arcs originating at a node in A′ and terminating at a node in A with $C' < 0$ and flow greater than or equal to the lower bound.

Since C′ can be calculated for every arc in sets B and B′, one should proceed as follows:

1. Case 1: For any $C' > 0$

   Define $\zeta_1 = \min_{B} [C'_{xy}]$ if $B \neq \varnothing$; otherwise, $\zeta_1 = \infty$.

2. Case 2: For any $C' < 0$

   Define $\zeta_2 = \min_{B'} [C'_{xy}]$ if $B' \neq \varnothing$; otherwise, $\zeta_2 = \infty$.

3. Let $\zeta = \min [\zeta_1 , \zeta_2]$.

4. For all node numbers (Π values) in the set A', add ζ to every Π_k, where k is a member of the set A'.
5. Do not erase any previous labels.

The process then continues by returning to the labeling procedure.

4.6 Summary and Conclusion

The theory of the out-of-kilter algorithm has been discussed in a heuristic setting, using the well-known theory of dual linear programming. All the steps leading to an optimal solution have been examined, and a logical explanation was attempted to justify why those steps were taken. A table has been presented summarizing the various changes that are possible as the algorithm proceeds, along with step-by-step procedures to be employed when attacking the problem. The network is established and an initial circulation is chosen which satisfies the conservation of flow. A circulation of zero will always satisfy this condition. The power of the out-of-kilter algorithm is characterized by simple manipulation and fast convergence. Actually, the algorithm may initiate with any set of flows that satisfies conservation of flow. In the next chapter, the Fault Dictionary is implemented into a network flow model. By applying the OKA, a feasible solution can be found.

5. Apply Mathematical Method to minimize test sets

5.1 Implementation of OKA to the Fault Dictionary

For a circuit with m test vectors and n distinct number of possible stuck-at-faults, we can obtain a Fault Dictionary as described in Chapter 3. With that information, we can transform it into a network model as shown in Figure 5.1.

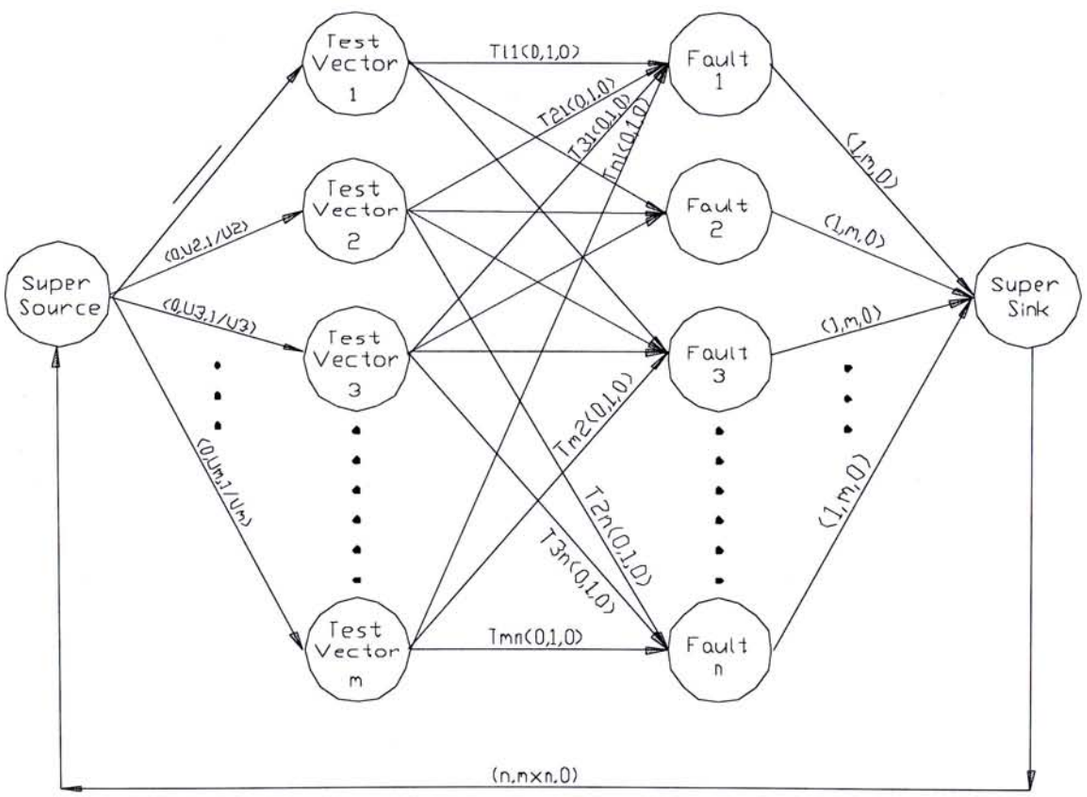


Figure 5.1 Fault Dictionary transform to Network Flow Algorithm

$$\text{Min} \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} f_{ij}$$

subject to

Conservation of flow

$$\forall i = 1,2,.....,m, \quad \sum_{j=1}^{n} f_{ji} - \sum_{j=1}^{n} f_{ij} = 0$$

Lower bounds

$$\forall i = 1,2,.....,m,$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \geq L_{ij}$$

Upper bounds

$$\forall i = 1,2,.....,m,$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \leq U_{ij}$$

Non negative flow

$$\forall i = 1,2,.....,m,$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \geq 0$$

All faults must be tested once

$$\forall j = 1,2,.....,n, \quad \sum_{i=1}^{m} T_{ij} \geq 1$$

Test vectors are either selected or not selected

$$\forall i = 1,2,.....,m, \quad \sum_{j=1}^{n} T_{ij} = 0 \text{ or } U_i$$

$$\text{Min} \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} f_{ij}$$

subject to

Conservation of flow

$$\forall i = 1,2,.....,m, \quad \sum_{j=1}^{n} f_{ji} - \sum_{j=1}^{n} f_{ij} = 0$$

Lower bounds

$$\forall i = 1,2,.....,m,$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \geq L_{ij}$$

Upper bounds

$$\forall i = 1,2,.....,m,$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \leq U_{ij}$$

Non negative flow

$$\forall i = 1,2,.....,m,$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \geq 0$$

All faults must be tested once

$$\forall j = 1,2,.....,n, \quad \sum_{i=1}^{m} T_{ij} \geq 1$$

Test vectors are either selected or not selected

$$\forall i = 1,2,.....,m, \quad \sum_{j=1}^{n} T_{ij} = 0 \text{ or } U_i$$

It is observed that the super source is a reservoir and it generates test vectors. Every test vector has it's own characteristic and the number of faults being detected by this test vector is represented by the arcs from the test vector to the faults. In order to keep a balance flow, all the faults are collected at the super sink and brought back to the super source by a return arc.

The number of test vectors depends on the Fault Dictionary. As mentioned in Chapter 3, an exhaustive method is being used. It implies a circuit with x input, the Fault Dictionary will consist of 2^x number of test vectors. For a circuit with y internal nodes, there exist a stuck-at-zero and a stuck-at-one fault occurring at each node. Therefore, a total number of 2y faults will be presented inside the network model.

Consider the capacitated arc $(0, U, \frac{1}{U})$ from the super source to the test vectors. The lower bound is set at 0 and the upper bound U depends on the number of faults being tested by that test vectors. Since there is no preference to select any of the test vectors, a cost C will be set at $\frac{1}{U}$ to maintain the same cost of one.

For the capacitated arc (0,1,0) from the test vectors to the faults, the lower bound and upper bound are set at 0 and 1 respectively. It can be understood that the test vector will be either selected or not selected. If the test vector is selected, the arcs will be sitting at the upper bound with cost at zero. While the test vector is not selected, the arcs will be sitting at the lower bound with zero cost.

The capacitated arc (1,m,0) from the stuck-at-faults to the super sink are set with lower bound at one so that every stuck-at-fault must be tested at least once. The upper bound is to assume all m test vectors can detect that stuck-at-faults. And the cost is also set at zero.

The return arc from super sink to super source are (n,mxn,0). The lower bound is n since all n stuck-at-faults must be tested at least once. The upper bound mxn assume all m test vectors can test n stuck-at-faults. The cost is set at zero.

5.2 Minimize test set and optimize internal probings/probe points

In order to arrive at a minimum test set, we need to extract information from a Fault Dictionary which takes all possibilities into consideration. Apply kilter 1 and kilter 2 algorithm in Chapter 5.2.1, we get a solution set which holds all the minimum test sets. In Chapter 5.2.2/Chapter 5.2.3, all the test sets are checked through the optimization algorithm, the minimum solution set which is the minimum test set with optimum number of internal probings/probe points is deduced.

5.2.1 Minimize the number of test vectors

After the formulation of the Fault Dictionary into a network flow model in which all test vectors are selected, it is obvious that the task now is to minimize the number of flow generated from the super source to the test vectors. It costs one unit for every generation of test vector from the super source. It is well known that nine kilter states occur (Refer to Table 4.1). There are three in-kilter states and six out-of-kilter states. The out-of-kilter states are all at α states. We start to put the most out-of-kilter arcs back to the in-kilter state. The cost per unit flow is higher for the test vector which can detect a smaller number of stuck-at-faults. Since the cost is $\frac{1}{U}$ where U is the total number of stuck-at-fault being detected.

Consider the flow change and select the most out-of-kilter arc by labeling technique, if a breakthrough occurs, all flows change state and the flow change continues. Figure 5.2 shows the flow chart of kilter 1 algorithm which does a quick scan of the test vectors and push arcs to in-kilter state as much as possible.

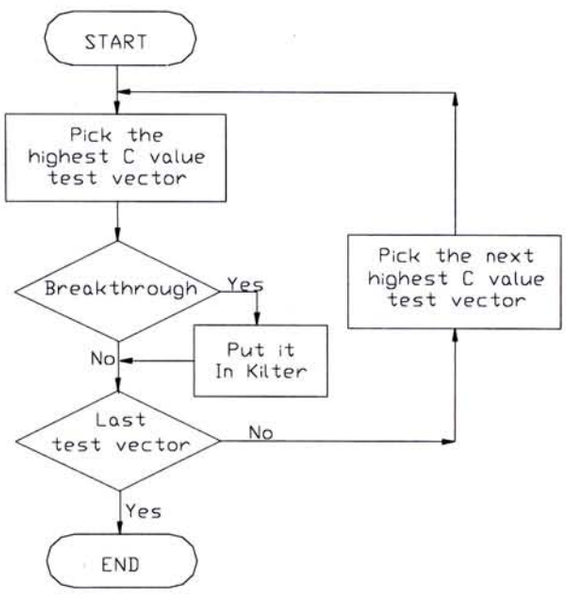


Figure 5.2 Flow chart of kilter 1 algorithm

For the remaining out-of-kilter arcs which arrive at a non-breakthrough state, we can treat those test set as an intermediate solution. In order to further reduce the test set, we need to apply kilter 2 algorithm which replace some of the vectors by a smaller number of more effective test vectors. Before the execution, the cost C need to be re-formulated by a new cost C*. There are two cases how C* are re-formulated:

Case 1

For those test vectors in the intermediate solution. The exclusive faults will be those whose arcs to the super sink node have only one unit of flow. The new cost C* will be 1/U where U is the number of exclusive faults detected by the test vector concerned.

Case 2

For those test vectors not in the intermediate solution. If a test vector can detect the exclusive faults as defined in the previous case, its cost will be modified according to the number of exclusive faults detected. If none is detected, the cost C* will become infinite.

After getting a new C* value, we continue to use the labeling technique with $C' = C^* + \Pi_i - \Pi_j$ and apply the kilter 2 algorithm as shown in Figure 5.3 until all the arcs get into in-kilter state. Hence, the minimum test set can be obtained.

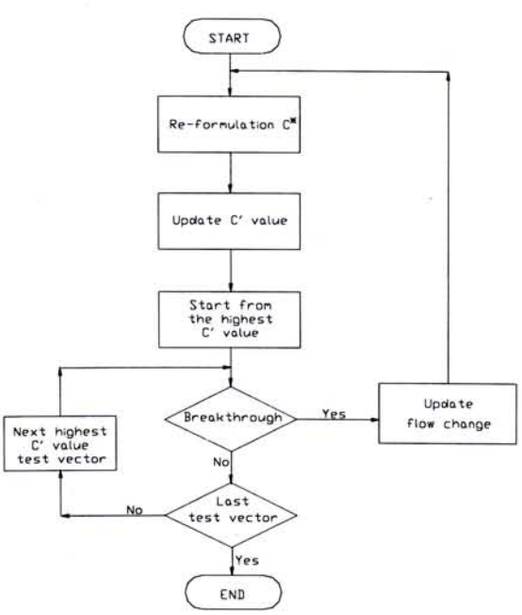


Figure 5.3 Flow chart of kilter 2 algorithm

5.2.2 Find the optimum number of internal probings

After getting m′ minimum number of test vectors where m′ < m from Chapter 5.2.1, the network model which shown m′ selected test vectors detecting n number of stuck-at-faults becomes:

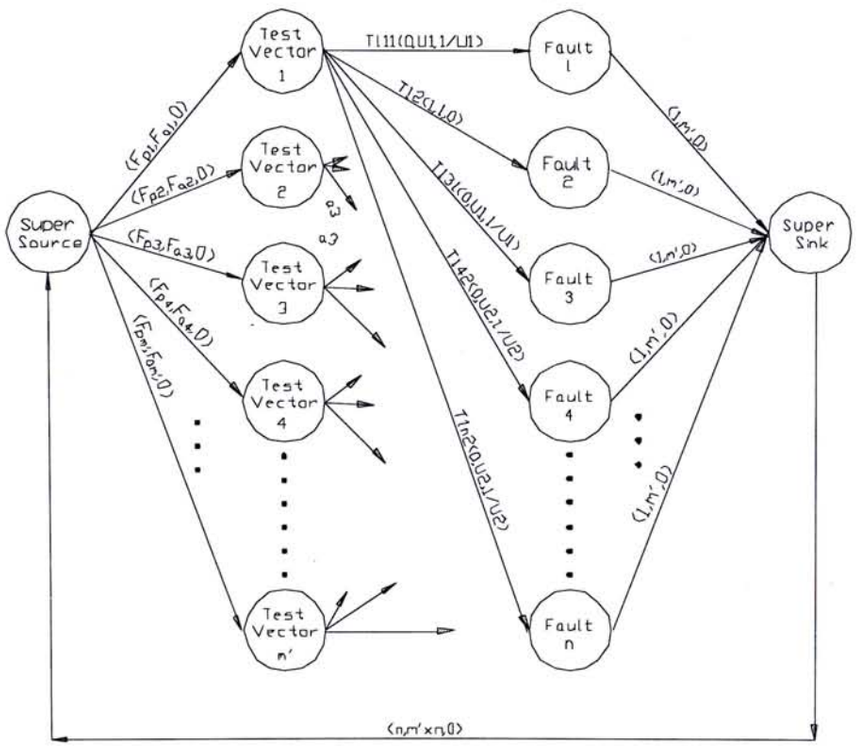


Figure 5.4 Network Model to optimize internal probing

For the capacitated arc from super source to test vector (F_{pm}, F_{am}, 0) shown in Figure 5.4, the lower bound F_{pm} represent the number of faults F detected by the primary output p of the test vector m and the upper bound F_{am} represent the number of faults F detected by the additional probing a of the test vector m. As our purpose is to delete redundant stuck-at-faults detected by additional probings.

The mathematical model becomes:

$$\text{Min} \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}f_{ij}$$

subject to

Conservation of flow

$$\forall i = 1,2,.....,m', \quad \sum_{j=1}^{n} f_{ji} - \sum_{j=1}^{n} f_{ij} = 0$$

Lower bounds

$$\forall i = 1,2,.....,m',$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \geq L_{ij}$$

Upper bounds

$$\forall i = 1,2,.....,m',$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \leq U_{ij}$$

Non negative flow

$$\forall i = 1,2,.....,m',$$
$$\forall j = 1,2,.....,n, \quad f_{ij} \geq 0$$

All faults must be tested once

$$\forall j = 1,2,.....,n, \quad \sum_{i=1}^{m'} T_{ij} \geq 1$$

Test vectors detecting fault by additional probings are either selected or not selected

$$\forall i = 1,2,.....,m',$$
$$\forall k = 1,2,.....,q, \quad \sum_{j=1}^{n} T_{ijk} = 0 \text{ or } U_{ik}$$

where k is the probing corresponding to that test vector.

For a selected test vector, we distinguish the fault being detected by primary output or internal probing by:

$T_{ij}(1,1,0)$ the capacitated arc which represent the fault detected by primary output of that test vector.

$T_{ijk}(0,U_{ik},\frac{1}{U_{ik}})$ the capacitated arc which represent the fault detected by k additional probing that corresponding test vector i.

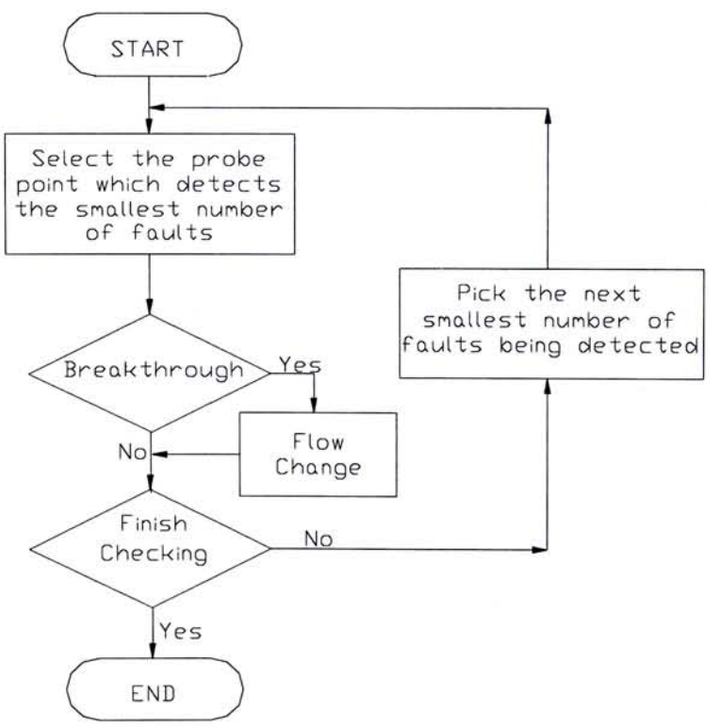


Figure 5.5 Flow chart of optimize algorithm

Figure 5.5 shows the detailed execution procedure of the optimization algorithm, where the most out-of-kilter state which has less number of faults detected by an additional point will be pushed to in-kilter first. Potential change is also applied to change the C′ value in order to push the arcs in-kilter.

5.2.3 Find the optimum number of internal probe points

The working procedure for finding the optimum number of internal probings is similar to the optimum number of internal probe points except the constraint:

Test vectors detecting fault by additional probings are either selected or not selected

$\forall i = 1,2,.....,m'$,

$\forall k = 1,2,.....,q, \quad \sum_{j=1}^{n} T_{ijk} = 0 \text{ or } U_{ik}$

where k is the probing corresponding to that test vector.

will be changed to

Test vectors detecting fault by additional probings are either selected or not selected

$\forall k = 1,2,.....,q, \quad \sum_{i=1}^{m'} \sum_{j=1}^{n} T_{ijk} = 0 \text{ or } U_{ik}$

where k is the probe point at the signal line inside the circuit.

and

$T_{ijk}(0, U_{ik}, \frac{1}{U_{ik}})$ becomes the capacitated arc which represents the fault detected by additional probe point at the line k of that circuit.

5.3 Fixed number of internal probings/probe points

This algorithm can find the minimum number of test vectors by fixing the number of internal probings/probe points.

The mathematical model is slightly changed by adding a constraint to the Mathematical model.

$\forall i = 1,2,.....,m,$

$\forall j = 1,2,.....,n, \quad \sum_{i=1}^{m} \sum_{j=1}^{n} T_{ijk} \leq p$

where p is the fixed number of additional probing/probe point being chosen.

The procedure simply starts as described in Chapter 5.2.1 and gets through kilter 1 algorithm. In kilter 2 algorithm, it is checked from time to time by the optimization algorithm as described in Chapter 5.2.2 / Chapter 5.2.3. If the above constraint is not being satisfied, breakthrough cannot be occurred. The algorithm is then looped back to kilter 2 algorithm again and the detailed procedure is shown in Figure 5.6.

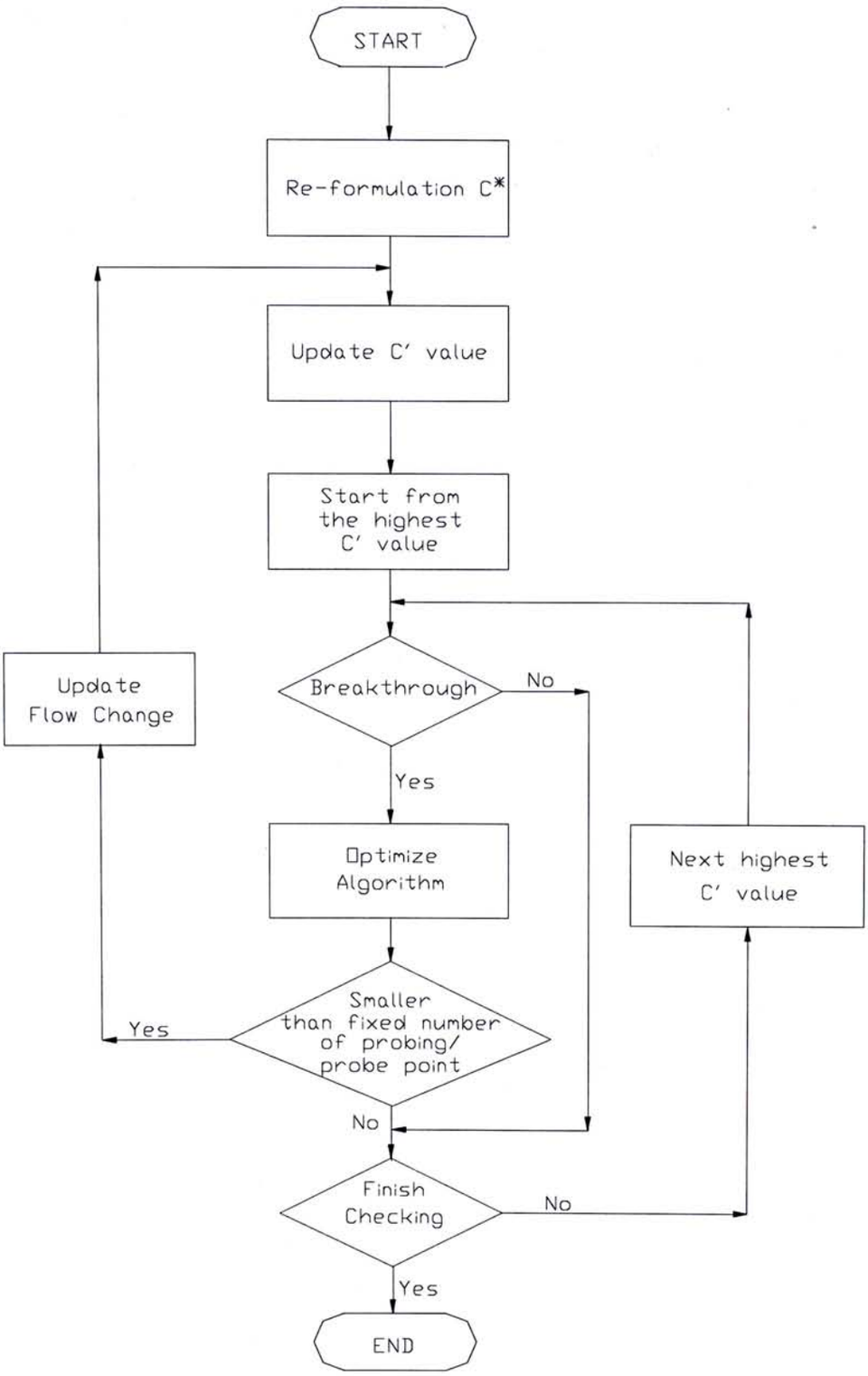


Figure 5.6 Flow chart to obtain minimum test set with fixed number of internal probings/probe points

5.4 True minimum test set and optimum probings/probe points

After getting a solution from Chapter 5.2, we need to consider if there exist an equivalent solution which has less probings/probe points. In order to avoid this situation to occur, we can safely check by running the algorithm again with fixed probings/probe points of the solution set. If the same result happen, it implies that it is the true minimum. If there exist a better solution set, we continuous to set the fixed probings/probe points equal to the previous number minus one until an equivalent solution which implies the true minimum. A detail procedure is shown in Figure 5.7:

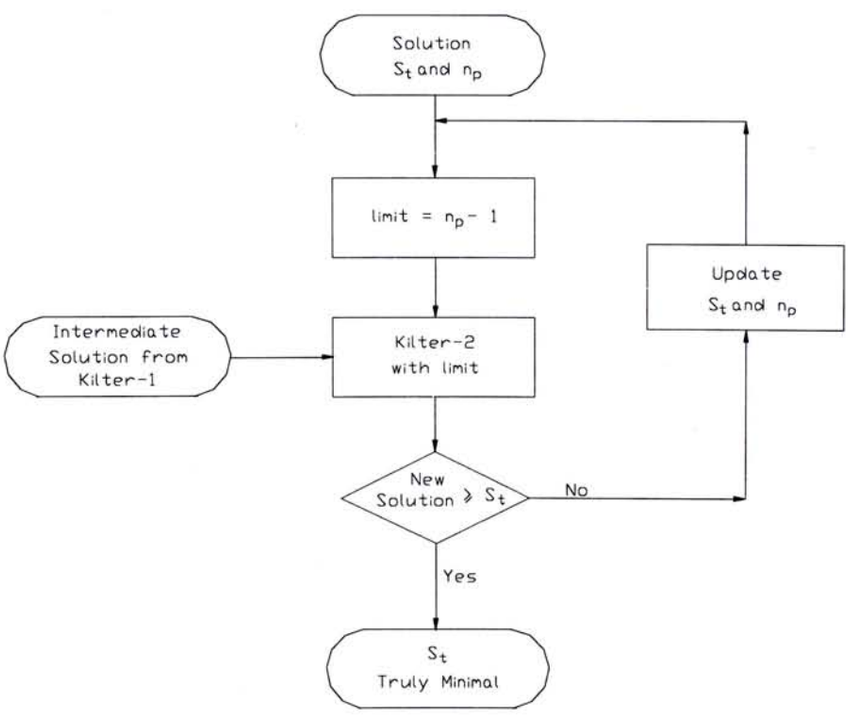


Figure 5.7 Flow chart to ensure a true minimum solution

6. Implementation and work examples

Throughout this chapter, we make use of the circuit SC7 to carry out the implementation and work examples. Except in Chapter 6.2 which shows the minimum test set without internal probe point, we make use of circuit SC1. It is because circuit SC1 is a better example in that section and it shows more detail steps than the circuit SC7.

6.1 Generation of Fault Dictionary

Consider the logic module in Figure 6.1,

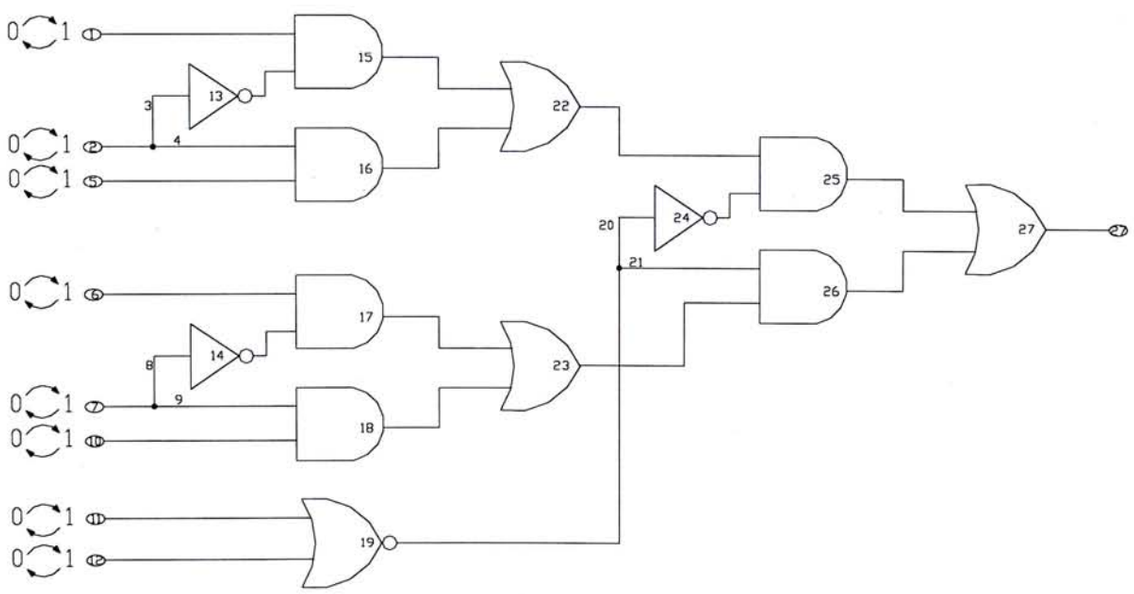


Figure 6.1 Circuit SC7 - A logic module

We start the execution of the following algorithm steps:

Step 1 : Read the circuit description of the circuit SC7.

Step 2 : Get the first test vector (0,0,0,0,0,0,0,0) to the primary input.

Step 3 : Execute fault-free simulation.

Step 4 : Critical path tracing from the primary input to the primary output.

Breakthrough : Mark '8' & '9' to represent stuck-at-1 and stuck-at-0 detected by primary output.

Non-Breakthrough : Add a probe point to the output of the non-breakthrough gate. Mark '4' & '5' to represent stuck-at-1 and stuck-at-0 detected by this probe point.

Step 5 : Get another test vector and back to step 2 until the last test vector is implemented.

Considering circuit SC7 and we have the test vector (0,1,0,1,1,0,1,0) at step 2. Figure 6.2 shows the execution of step 2 to step 4.

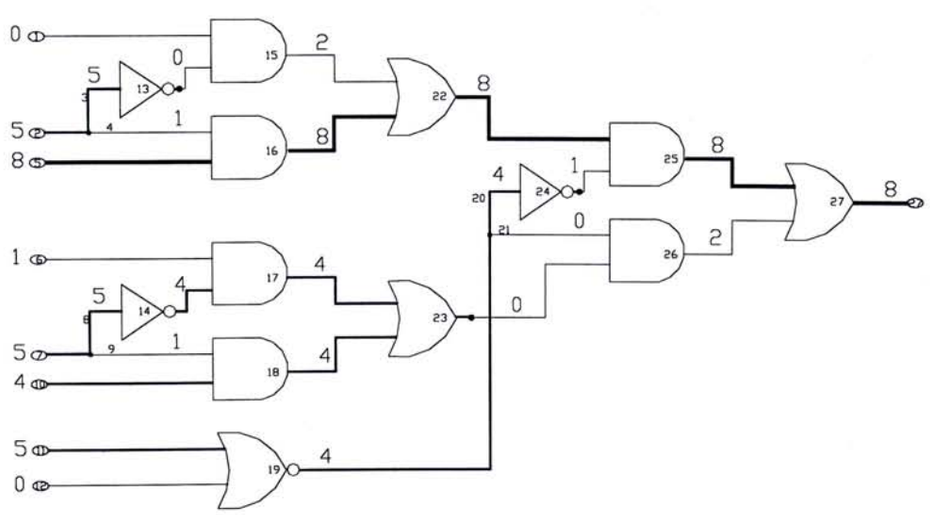


Figure 6.2 Logic state of Circuit SC7 with input test vector (0,1,0,1,1,0,1,0)

The element "91" of the Fault Dictionary becomes

```
055181551450042844440801828-3+90*
.000000550400040044000000000-23+6*
.000000000050000000440000000-24+3*
.055000000000000000000000000-13+2*
```

The first line shows all the logic value of the circuit, and Path (5,16,22,25,27) is those stuck-at-one faults at lines 5, 16, 22, 25 & 27 detected by the primary output. '-3' indicate that there are at most 3 probe points for this test vector. And '90' identify the value of this test vector coded in decimal value. There are totally 0 - 255 numbers of test vectors.

The second line indicates stuck-at-one faults at lines 10,14,17&18 and stuck-at-zero faults at lines 7&8 by the probe point at line 23. '+6' shows that six faults are detected by this probe point. The third and fourth lines are similar.

Repeating the above algorithm steps until all 256 test vectors are evaluated. We have the Fault Dictionary as shown in Appendix B.

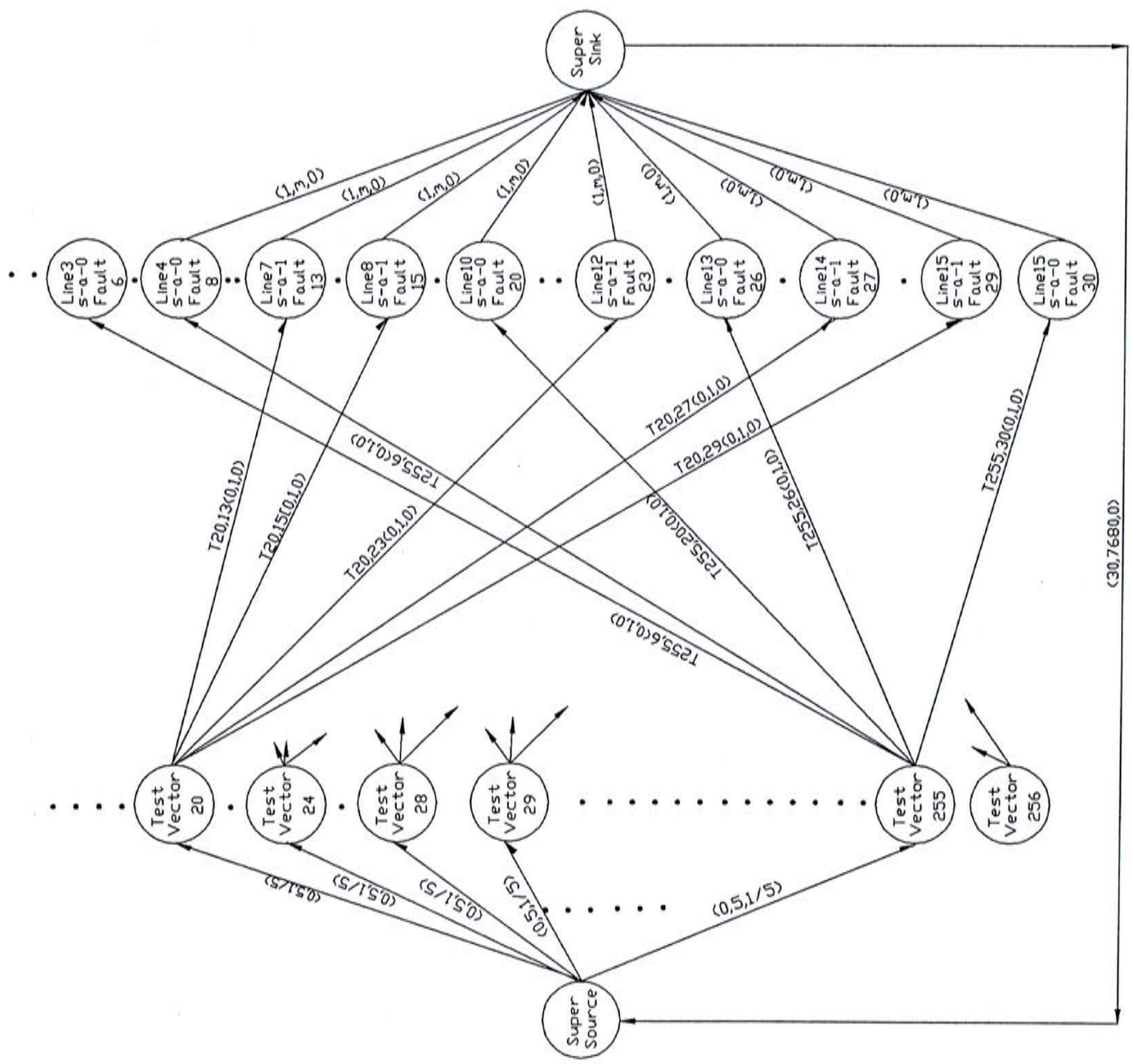


Figure 6.3 Network model of circuit SC1 with initial state by conventional method

6.2 Finding the minimum test set without internal probe point

For the conventional method, we only consider those stuck-at-faults detected by the primary output. Hence, we consider only the value '8' and '9' in the Fault Dictionary and ignore '0-5'. In this example, we pick circuit SC1 instead of circuit SC7 and the network model is shown in Figure 6.3. The initial state is set up with the information of the Fault Dictionary. Go to kilter 1 algorithm,

Step 1 : Pick the most out-of-kilter arc. i.e. high C value.

Step 2 : Check the flow change, Is breakthrough occur?

Yes, put the arc in kilter.

No, mark this arc, go to the next vector.

Step 3 : Go to step 1, until all the test vector is checked.

After kilter 1 algorithm, we have seven test vectors which are out-of-kilter and the rest, 249 test vectors, are in-kilter. We continue to push those out-of-kilter arcs into kilter and consider the new C* value. The status are shown as below:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
64	5	1/5	1/3
111	10	1/10	1/1
159	10	1/10	1/1
172	10	1/10	1/3
208	5	1/5	1/3
247	6	1/6	1/2
250	6	1/6	1/2

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
1	0	∞	∞
:	:	:	:
:	:	:	:
100	10	1/10	1/4
:	:	:	:
:	:	:	:
256	0	∞	∞

Consider the potential change in Chapters 4.5 and 5.2.1, the new C* and C′ = $C^*+\Pi_i-\Pi_j$ with $\Pi i=0$, $\Pi_j=1/4$:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
64	5	1/5	1/3	1/12
111	10	1/10	1/1	3/4
159	10	1/10	1/1	3/4
172	10	1/10	1/3	1/12
208	5	1/5	1/3	1/12
247	6	1/6	1/2	1/4
250	6	1/6	1/2	1/4

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
1	0	∞	∞	∞
:	:	:	:	:
:	:	:	:	:
100	10	1/10	1/4	0
:	:	:	:	:
:	:	:	:	:
1	0	∞	∞	∞

With the new C' value, breakthrough occurs. The arc (S,T100) is pushed to upper bound and the arc (S,T111)&(S,T172) are pushed to lower bound. Now, the status becomes:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
64	5	1/5	1/3
100	10	1/10	1/5
159	10	1/10	1/5
208	5	1/5	1/3
247	6	1/6	1/2
250	6	1/6	1/2

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
1	0	∞	∞
:	:	:	:
:	:	:	:
:	:	:	:
:	:	:	:
256	0	∞	∞

With further changing of the C′ value by changing Π_j=1/5, the arc (S,T100)& (S,T159) become in kilter without any breakthrough occurring. Continue changing Π_j=1/3 and Π_j=1/2 bring all the arcs in kilter. The optimum solution is:

555511881118188-2+64
990009909099999-0+100
419990090999999-1+159
555588111181188-2+208
811811118833818-0+247
188111118833818-0+250

We need six test vectors to test all the stuck-at-faults in the circuit from the primary output and they are 64, 100, 159, 208, 247 and 250.

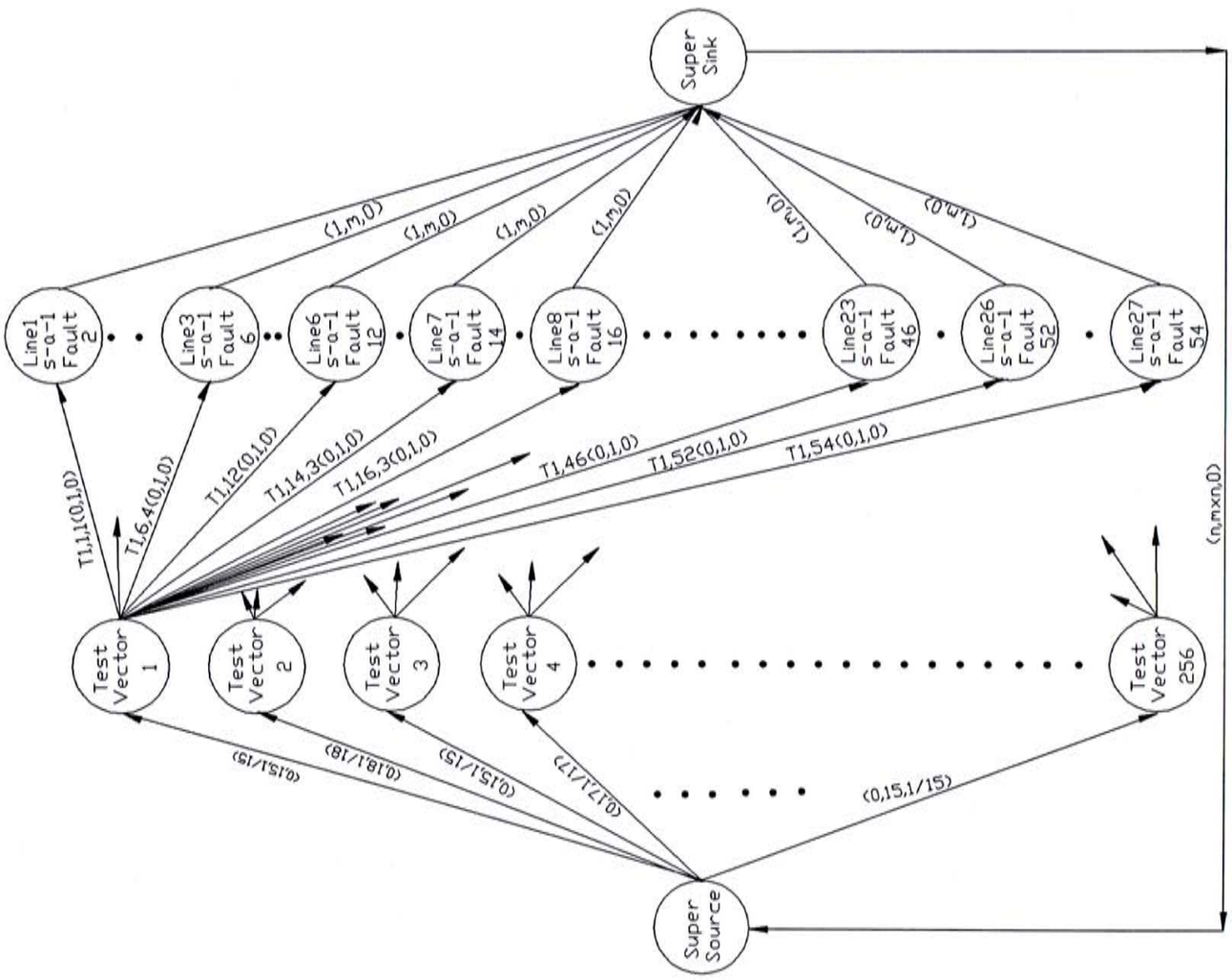


Figure 6.4 The initial state of Network model for circuit SC7 with additional probing

6.31 Finding the minimum test set with optimum internal probing

With the increasing observability by additional internal probings, we consider the value '4' & '8' for stuck-at-one faults and the value '5' & '9' for stuck-at-zero faults. Obviously, the number of test vectors needed should be less comparing with conventional method. Regarding the circuit SC7, we formulate it as a network flow model as shown in Figure 6.4. The initial state is set up with the information of the Fault Dictionary.

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value
1	15	1/15
2	18	1/18
:	:	:
:	:	:
:	:	:
255	14	1/14
256	15	1/15

Go to kilter 1 algorithm,

Step 1 : Pick the most out-of-kilter arc. i.e. high C value.

Step 2 : Check the flow change, does breakthrough occur?

Yes, put the arc in kilter.

No, mark this arc, go to the next vector.

Step 3 : Go to step 1, until all the test vectors are checked.

After kilter 1 algorithm, we have eight test vectors which are out-of-kilter and the rest, 248 test vectors, are in-kilter. We continue to push those out-of-kilter arcs into kilter and consider the new C* value. The status are shown as below:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
32	21	1/21	1/1
40	21	1/21	1/1
48	21	1/21	1/3
64	21	1/21	∞
128	20	1/20	1/1
188	20	1/20	1/1
189	20	1/20	1/1
190	20	1/20	1/3

Since the kilter 1 is a quick scan algorithm, it needs to reach a state that no additional probing is necessary before going to kilter 2 algorithm. It is the reason why there is a chance for the number of effective fault equal to zero and making C* to infinity.

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
1	15	1/15	1/2
:	:	:	:
:	:	:	:
202	13	1/13	1/4
:	:	:	:
:	:	:	:
256	15	1/15	∞

Consider the potential change in Chapters 4.5 and 5.2.1, the new C* and C′ = $C^*+\Pi_i-\Pi_j$ with $\Pi_i=0$, $\Pi_j=1/4$:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
32	21	1/21	1/1	3/4
40	21	1/21	1/1	3/4
48	21	1/21	1/3	1/12
64	21	1/21	∞	∞
128	20	1/20	1/1	3/4
188	20	1/20	1/1	3/4
189	20	1/20	1/1	3/4
190	20	1/20	1/3	1/12

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
1	0	1/15	1/2	1/4
:	:	:	:	:
:	:	:	:	:
202	13	1/13	1/4	0
:	:	:	:	:
:	:	:	:	:
256	0	∞	∞	∞

With the new C' value, breakthrough occurs. The arc (S,T202) is pushed to upper bound and the arcs (S,T48)&(S,T190) are pushed to lower bound. Now, the status becomes:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
32	21	1/21	1/1
40	21	1/21	1/2
64	21	1/21	1/3
128	20	1/20	1/2
188	20	1/20	1/1
189	20	1/20	1/2
202	13	1/13	1/5

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
1	15	1/15	1/2
:	:	:	:
:	:	:	:
90	18	1/18	1/6
:	:	:	:
:	:	:	:
256	15	1/15	∞

Consider the potential change in Chapters 4.5 and 5.2.1, the new C* and C′ = C*+Π_i-Π_j with Π_i=0, Π_j=1/6 :

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
32	21	1/21	1/1	5/6
40	21	1/21	1/2	1/3
64	21	1/21	1/3	1/6
128	20	1/20	1/2	1/3
188	20	1/20	1/1	5/6
189	20	1/20	1/2	1/3
202	13	1/13	1/6	1/30

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
1	0	1/15	1/2	1/3
:	:	:	:	:
:	:	:	:	:
90	18	1/18	1/6	0
:	:	:	:	:
:	:	:	:	:
256	0	1/15	∞	∞

With the new C' value, breakthrough occurs. The arc (S,T90) is pushed to upper bound and the arcs (S,T32)&(S,T128) are pushed to lower bound. With the new C* and C' = $C^*+\Pi_i-\Pi_j$, $\Pi_i=0$, $\Pi_j=1/4$. Now, the status becomes:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
40	21	1/21	1/2	1/4
64	21	1/21	1/3	1/12
90	20	1/18	1/3	1/12
188	20	1/20	1/1	3/4
189	20	1/20	1/2	1/4
202	13	1/13	1/3	1/12

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
1	0	1/15	1/3	1/12
:	:	:	:	:
:	:	:	:	:
204	18	1/18	1/4	0
:	:	:	:	:
:	:	:	:	:
256	0	1/15	∞	∞

With the new C' value, breakthrough occurs. The arc (S,T204) is pushed to upper bound and the arcs (S,T188)&(S,T202) are pushed to lower bound. With the new C* and C' = $C^*+\Pi_i-\Pi_j$, $\Pi_i=0$, $\Pi_j=1/8$. Now, the status becomes:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
40	21	1/21	1/3	5/24
64	21	1/21	1/3	5/24
90	20	1/18	1/8	0
189	20	1/20	1/5	3/40
204	13	1/14	1/4	1/8

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
1	0	1/15	1/3	5/24
:	:	:	:	:
:	:	:	:	:
204	18	1/18	1/7	1/56
:	:	:	:	:
:	:	:	:	:
256	0	1/15	1/2	3/8

With the new C' value, no breakthrough occurs. Π_j value is further changed to 1/7 and hence $C' = C^* + \Pi_i - \Pi_j$, $\Pi_i = 0$, $\Pi_j = 1/7$. Now, the status becomes:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
40	21	1/21	1/3	4/21
64	21	1/21	1/3	4/21
189	20	1/20	1/5	2/35
204	13	1/14	1/4	3/28

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
1	0	1/15	1/3	4/21
:	:	:	:	:
12	19	1/19	1/7	0
:	:	:	:	:
90	20	1/18	1/8	-1/56
:	:	:	:	:
:	:	:	:	:
256	0	1/15	1/2	5/14

With the new C' value, breakthrough occurs. The arc (S,T12) is pushed to upper bound and the arcs (S,T64)&(S,T204) are pushed to lower bound. The status are:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
12	19	1/19	1/7
40	21	1/21	1/7
90	18	1/18	1/8
189	20	1/20	1/8

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
1	0	1/15	1/3
:	:	:	:
:	:	:	:
:	:	:	:
:	:	:	:
:	:	:	:
256	0	1/15	1/8

Continue changing Π_j=1/8 and Π_j=1/7 brings all the arcs in kilter. The minimum number of test vectors are :

15514988008849449095909009 9-2+12
15555884818841058899118888 8-3+40
98800155149094904488090990 9-1+90
88481155550914880584881188 8-4+189

To optimize the number of internal probe point, we go to the optimization algorithm by formulating those vectors into a network flow model as shown in Figure 6.5.

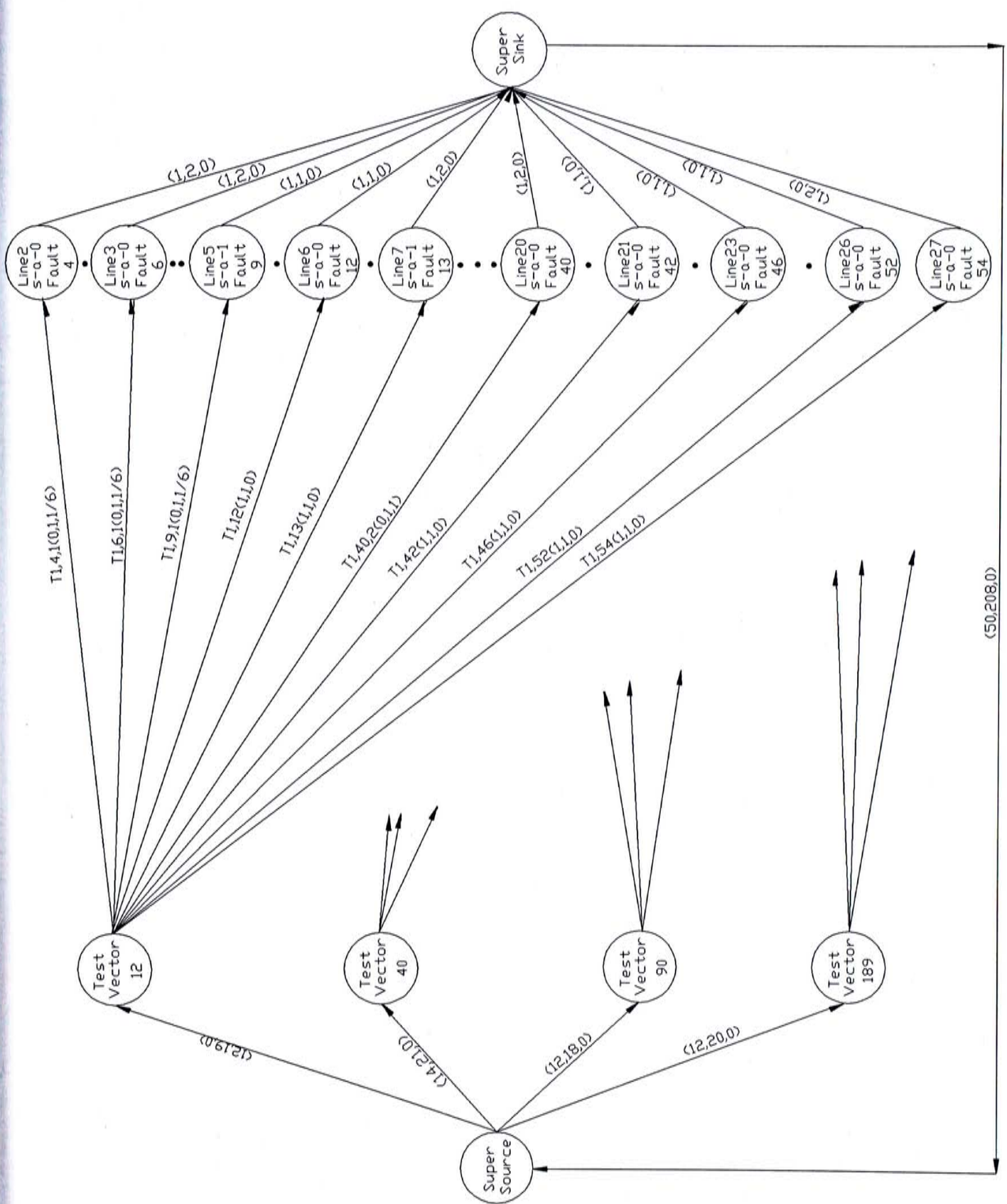


Figure 6.5 Network model to find optimize probing with minimum number of test vectors

The following table shows the effectiveness of the probing correspondance to the respective test vector. The larger the value of C, the less importance of that probe point correspondance to the test vector and therefore the most out-of-kilter represented in the network flow model.

Vector with probing	fault detect	C Value
vector 12 probe at 22	6	1/6
vector 12 probe at 24	1	1/1
vector 40 probe at 22	4	1/4
vector 40 probe at 15	3	1/3
vector 40 probe at 14	1	1/1
vector 90 probe at 23	6	1/6
vector 189 probe at 23	4	1/4
vector 189 probe at 24	1	1/1
vector 189 probe at 17	3	1/3
vector 189 probe at 13	1	1/1

By checking the most out-of-kilter state, i.e. C value = 1, four probings are in kilter by pushing them to lower bound. The status becomes:

Vector with probing	fault detect	C Value
vector 12 probe at 22	6	1/6
vector 40 probe at 22	4	1/4
vector 40 probe at 15	3	1/3
vector 90 probe at 23	6	1/6
vector 189 probe at 23	4	1/4
vector 189 probe at 17	3	1/3

Checking the next most out-of-kilter, i.e. C value = 1/3, two more probings are pushed to be in kilter. Now, there are only four probings which are optimized to the corresponding test vectors as shown:

Vector with probe point	fault detect	C Value
vector 12 probe at 22	6	1/6
vector 40 probe at 22	4	1/4
vector 90 probe at 23	6	1/6
vector 189 probe at 23	4	1/4

Futher checking by $C' = C+\Pi_i-\Pi_j$ and there is no more changes, therefore the optimum solution is:

```
1551498800884944909590900 99-2+12
.0550400000004044000000000 0-22+6
1555588481884105889911888 88-3+40
.0505500000000005000000000 0-22+4
9880015514909490448809099 09-1+90
.0000005504000400440000000 00-23+6
8848115555091488058488118 88-4+189
.0000005055000000050000000 00-23+4
```

In conclusion, we need four test vectors with four internal probings to test all the stuck-at-faults in the circuit. They are the vector 12 with probe at line 22, vector 40 with probe at line 22, vector 90 with probe at line 23 and vector 189 with probe at line 23.

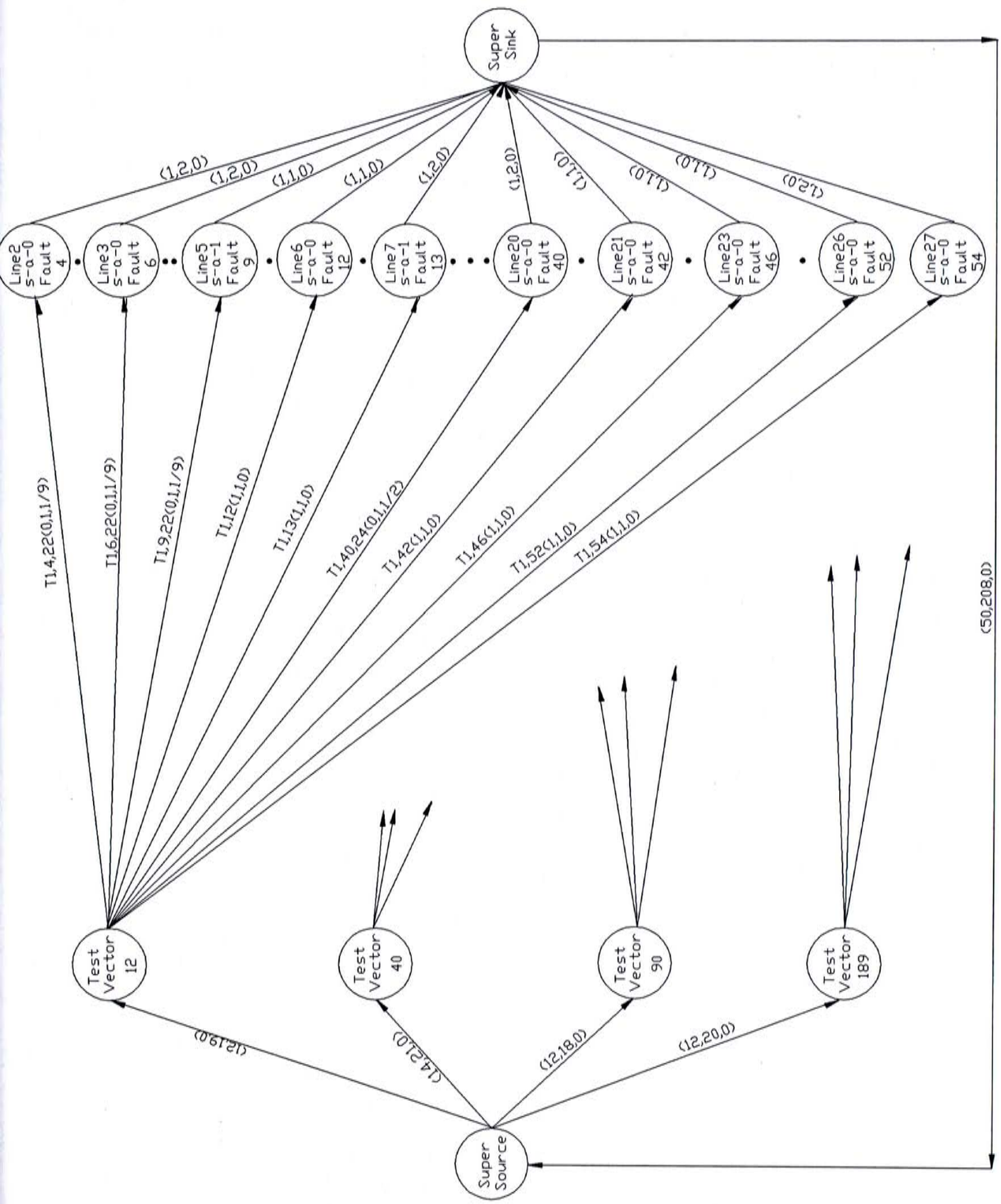


Figure 6.6 Network model to find optimize probe point with minimum number of test vectors

6.3.2 Finding the minimum test set with optimum internal probe point

Repeating the procedures in Chapter 6.31 and getting the minimum test set:

```
1551498800884944909590900 99-2+12
1555588481884105889911888 88-3+40
9880015514909490448809099 09-1+90
8848115555091488058488118 88-4+189
```

we formulate the problem as shown in Figure 6.6 and the table to show the effectiveness of the probe point corresponding to the respective vector becomes:

Vector with probe point	fault detect	C Value
vector 189 probe at 13	1	1/1
vector 40 probe at 14	1	1/1
vector 40 probe at 15	3	1/3
vector 189 probe at 17	3	1/3
vector 12,40 probe at 22	9	1/9
vector 90,189 probe at 23	9	1/9
vector 12,189 probe at 24	2	1/2

By checking the most out-of-kilter state, i.e. C value = 1, two probe points are in kilter by pushing them to lower bound. The status becomes:

Vector with probe point	fault detect	C Value
vector 40 probe at 15	3	1/3
vector 189 probe at 17	3	1/3
vector 12,40 probe at 22	9	1/9
vector 90,189 probe at 23	9	1/9
vector 12,189 probe at 24	2	1/2

Checking the next most out-of-kilter, i.e. C value = 1/2 & 1/3, two more probe points are pushed to be in kilter. Now, there are only two probe points which are optimized to the corresponding test vectors as shown:

Vector with probe point	fault detect	C Value
vector 12,40 probe at 22	9	1/9
vector 90,189 probe at 23	9	1/9

Futher checking by $C' = C+\Pi_i-\Pi_j$ and there is no more changes, therefore the optimum solution is:

```
1551498800884944909590900 99-2+12
.0550400000004044000000000 00-22+6
1555588481884105889911888 88-3+40
.0505500000000005000000000 00-22+4
9880015514909490448809099 09-1+90
.0000005504000400440000000 00-23+6
8848115555091488058488118 88-4+189
.0000005055000000050000000 00-23+4
```

In conclusion, we need four test vectors with two internal probe points to test all the stuck-at-faults in the circuit. They are the vector 12, 40, 90 and 189 with probe points at line 22 and 23.

6.4 Finding the minimum test set by fixing the number of internal probings at 2

We start with the network flow model same as shown in Figure 6.4. The initial state is set up with the information of the Fault Dictionary as shown:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
1	15	1/15	1/5
2	18	1/18	1/11
:	:	:	:
:	:	:	:
:	:	:	:
255	14	1/14	1/7
256	15	1/15	1/7

Go to kilter 1 algorithm,

Step 1 : Pick the most out-of-kilter arc. i.e. high C value.

Step 2 : Check the flow change, Is breakthrough occur?

Yes, put the arc in kilter.

No, mark this arc, go to the next vector.

Step 3 : Go to step 1, until all the test vector is checked.

After kilter 1 algorithm, we have eight test vectors which are out-of-kilter and the rest, 248 test vectors, are in-kilter. We continue to push those out-of-kilter arcs into kilter and consider the new C* value. The status are shown as below:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
32	21	1/21	1/1
40	21	1/21	1/1
48	21	1/21	1/3
64	21	1/21	∞
128	20	1/20	1/1
188	20	1/20	1/1
189	20	1/20	1/1
190	20	1/20	1/3

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
1	15	1/15	1/2
:	:	:	:
:	:	:	:
202	13	1/13	1/4
:	:	:	:
:	:	:	:
256	15	1/15	∞

Consider the potential change in Chapters 4.5 and 5.2.1, the new C* and C′ = $C^*+\Pi_i-\Pi_j$ with $\Pi_i=0$, $\Pi_j=1/4$:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
32	21	1/21	1/1	3/4
40	21	1/21	1/1	3/4
48	21	1/21	1/3	1/12
64	21	1/21	∞	∞
128	20	1/20	1/1	3/4
188	20	1/20	1/1	3/4
189	20	1/20	1/1	3/4
190	20	1/20	1/3	1/12

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
1	0	1/15	1/2	1/4
:	:	:	:	:
:	:	:	:	:
202	13	1/13	1/4	0
:	:	:	:	:
:	:	:	:	:
256	0	1/15	∞	∞

With the new C' value, breakthrough occurs. The arc (S,T202) is pushed to upper bound and the arcs (S,T48)&(S,T190) are pushed to lower bound. Now, the status becomes:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
32	21	1/21	1/1
40	21	1/21	1/2
64	21	1/21	1/3
128	20	1/20	1/2
188	20	1/20	1/1
189	20	1/20	1/2
202	13	1/13	1/5

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
1	15	1/15	1/2
:	:	:	:
:	:	:	:
90	18	1/18	1/6
:	:	:	:
:	:	:	:
256	15	1/15	∞

Consider the potential change in Chapters 4.5 and 5.2.1, the new C* and C′ = $C^*+\Pi_i-\Pi_j$ with $\Pi_i=0$, $\Pi_j=1/6$:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
32	21	1/21	1/1	5/6
40	21	1/21	1/2	1/3
64	21	1/21	1/3	1/6
128	20	1/20	1/2	1/3
188	20	1/20	1/1	5/6
189	20	1/20	1/2	1/3
202	13	1/13	1/5	1/30

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
1	0	1/15	1/2	1/3
:	:	:	:	:
:	:	:	:	:
90	18	1/18	1/6	0
:	:	:	:	:
:	:	:	:	:
256	0	1/15	∞	∞

With the new C' value, breakthrough cannot occur. Since it needs to pass through the optimization algorithm. The result comes back is "three" internal probe points that are necessary for the test. It contradicts the constraint and hence no breakthrough occurs.

Now, the status becomes:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
32	21	1/21	1/1
40	21	1/21	1/2
64	21	1/21	1/3
128	20	1/20	1/2
188	20	1/20	1/1
189	20	1/20	1/2
202	13	1/13	1/5

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
1	15	1/15	1/2
:	:	:	:
:	:	:	:
220	15	1/15	1/2
:	:	:	:
:	:	:	:
256	15	1/15	∞

Consider the potential change in Chapters 4.5 and 5.2.1, the new C* and C′ = $C^*+\Pi_i-\Pi_j$ with $\Pi_i=0$, $\Pi_j=1/2$:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
32	21	1/21	1/1	1/2
40	21	1/21	1/2	0
64	21	1/21	1/3	-1/6
128	20	1/20	1/2	0
188	20	1/20	1/1	1/2
189	20	1/20	1/2	0
202	13	1/13	1/5	-3/10

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value	C' Value
1	0	1/15	1/2	0
:	:	:	:	:
:	:	:	:	:
220	15	1/15	1/2	0
:	:	:	:	:
:	:	:	:	:
256	0	1/15	∞	∞

With the new C' value and checking the optimization algorithm which results in "two" internal probe points that are necessary for the test. It satisfies the constraint and hence breakthrough occurs. The arc (S,T220) is pushed to upper bound and the arcs (S,T32)&(S,T188) are pushed to lower bound. Now, the status becomes:

Out of kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
40	21	1/21	1/3
64	21	1/21	1/3
128	20	1/20	1/2
189	20	1/20	1/3
202	13	1/13	1/5
220	15	1/15	1/2

In kilter test Vectors

Test Vector	No. of faults detected	C Value	C* Value
1	15	1/15	:
:	:	:	:
:	:	:	:
:	:	:	:
:	:	:	:
256	15	1/15	:

With the new C' value, the arc (S,T10) can be pushed to upper bound and the arcs (S,T64)&(S,T202) can be pushed to lower bound but it violates the constraint that the number of internal probe points equals to "three" after the optimization algorithm. Similar case happens to the arc (S,T74) which can be pushed to upper bound and the arcs (S,T128)&(S,T202) can be pushed to lower bound but it violates the constraint that the number of internal probe points equals to "three" after the optimization algorithm.

After checking all the test vectors, the optimum solution becomes:

```
1555588481884105889911888888-3+40
1555519599884405099591940 99-4+64
1959915555904409058849199 09-4+128
8848115555091488058488118 88-4+189
9880054400119590502229139 09-1+202
.0000054400000500500000000 00-23+5
1991815514118488442208018 28-1+220
.0000005504000400440000000 00-23+6
```

In conclusion, we need six test vectors with two internal probe points to test all the stuck-at-faults in the circuit. They are the vector 40, vector 64, vector 128, vector 189, vector 202 with probe at line 23 and vector 40 with probe at line 23.

6.5 Program description

This algorithm is written in C Language and run on the IBM personal computer which is a 486DX 33mHz with 8M Ram. The size of the program is 30 Kbytes for generating of the Fault Dictionary and 20 Kbytes for optimization. The running time of several simple circuits can be summarizes in Table 6.1.

Circuits	SC1	SC2	SC3	SC7
Running time for Generation of Fault Dictionary (Seconds)	6	9	130	10
Running time for optimization of test set and probings/probe points (Minutes)	3	2.3	480	4.5

Table 6.1 Running time for the simple circuits

7. Realistic approach to find the minimum solution

7.1 Problem arising in exhaustive method

The approach of the algorithm is to put the arc of the test vector from the out-of-kilter state to in-kilter state and the computational time is very much depend on the number of test vectors in the Fault Dictionary. However, a true minimum solution can only be obtained if all the possibility are put into consideration and hence every test vector is involved.

Considering a combinational circuits with 8 primary inputs, it has 2^8 = 256 number of different input test vectors. A 16 input circuits has 65536 different input test vectors while a 32 input circuits has 4294967296 different input test vectors. Now, we need to think about the advantage of selecting all comparing with partially selection. In Chapter 7.2, we can observe from an exhaustive method the weakness of an existing test generation algorithm. So we can suggest two more methods at Chapter 7.3 to reduce the search set. With a smaller Fault Dictionary which holds only effective test vectors, the time and cost of applying the algorithm can be much reduced.

7.2 Improvement work on existing test generation algorithm

Base on the optimization work for simple circuits, we compare the result obtained by the existing algorithm [7,29] to the results obtained by the out-of-kilter algorithm as shown in Table 7.1.

Circuit	Test generation by existing method	Test generation by out-of-kilter algorithm
SC1	4 test vector & 6 probing	4 test vector & 4 probing
SC2	4 test vector & 5 probing	4 test vector & 4 probing
SC3	9 test vector & 7 probing	5 test vector & 9 probing

Table 7.1 Result of comparison between an existing algorithm and the out-of-kilter algorithm

We can draw some conclusions to improve the existing algorithm to arrive at the optimum solution.
Consider the following procedure for the existing algorithm:

Step 1 : Read the circuit description file.
Step 2 : Build a fault list.
Step 3 : Select a seed line.
Step 4 : Apply propagation and justification rules:
A critical path is generated forward towards the POs.
A critical path is generated backward towards the PIs.

Potential probings are added and justification is carried out from these points.
Then, a test vector with corresponding probings are generated.
The stuck-at-faults detected by this test vector are also obtained.

Step 5 : Remove redundant probings.
Then, probings and detected faults are revised.

Step 6 : Remove detected faults from the fault list.

Step 7 : Repeat from step 3 if the fault list is not empty.

Step 8 : A set of test vectors with corresponding probings and faults detected are generated.

With the optimization solution, we can deduce a better test generation by considering the following two points,

1.Consider a second seed line instead of only one seed line.
2.Remove redundant probing at the latest stage.

If we modify the algorithm as shown below:

Step 1 : Read the circuit description file.

Step 2 : Build a fault list.

Step 3 : Select a seed line.

Step 4 : Select a back up seed line.

Step 5 : Apply propagation and justification rules:
A critical path is generated forward towards the POs with the seed line.
A critical path is generated forward towards the POs with the back up seed line.
If confliction exists, add internal probing.
A critical path is generated backward towards the PIs.
Potential probings are added and justification is carried out from these points.
Then, a test vector with corresponding probings is generated.
The stuck-at-faults detected by this test vector are also obtained.

Step 6 : Remove detected faults from the fault list.

Step 7 : Repeat from step 3 until the fault list is empty.

Step 8 : Remove redundant probings.

Step 9 : A minimum set of test vectors with optimum number of probings is generated.

The advantage of considering another seed line instead of only one seed line makes circuit SC3 arrives at the known minimum of 5 vectors with 9 probings. And the benefit of considering the redundant probing at the last stage, circuit SC1 arrives at the known minimum of 4 test vectors with 4 probings while circuit SC2 is optimized to 4 vectors with 4 probings.

7.3 Reduce the search set

It is obvious that there are a lot of redundant test vectors in the Fault Dictionary. They will require a lot of memory area and hence computational time. We start to think about the possibility of having a quick selection method to reduce the search set, hence to reduce computational time and saving cost. There are at least two possible ways to reduce the search set.

7.3.1 Making the Fault Dictionary from existing test generation algorithm

The Fault Dictionary can be generated by exploiting the frequent occurrence of arbitrary choices in an existing test generation algorithm. In the algorithm [7], when there is a choice, it always puts the signal line with the highest number critical. Such a blind solution will certainly not guarantee an optimum result. Let's consider the first iteration of the existing algorithm with circuit SC2 as shown in Figure 7.1:

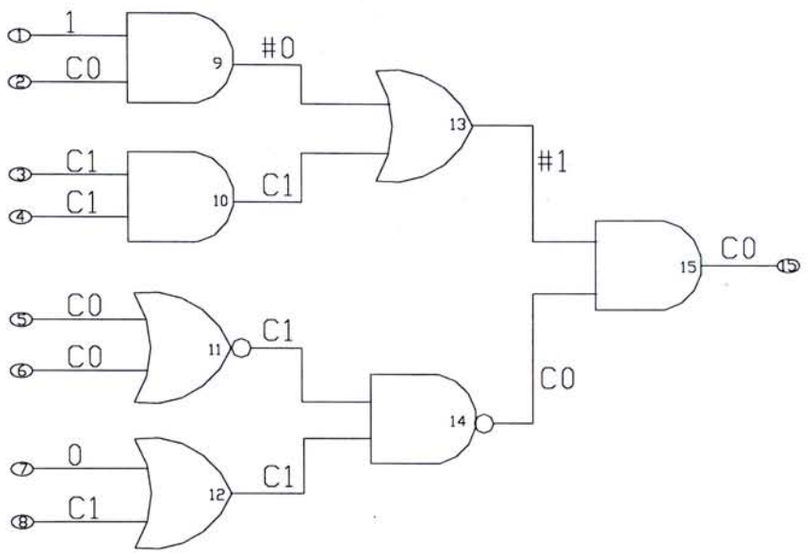


Figure 7.1 First generation for Circuit SC2

The first iteration puts critical C1 at line 8 but we can put C1 at line 7 instead of line 8. If we consider this alternative, it may help us to arrive at the optimum after minimization. The test vectors added to the Fault Dictionary are:

1455880599188-2+141 1455890599188-2+141

.005500000500000-13+3 and .005500000500000-13+3

.040000000000000-9+1, .040000000000000-9+1

Swapping lines 1 & 2 and also lines 9 & 10 give 4 possible test vectors. A total 4 x 2 = 8 test vectors will be considered. In Figure 7.2, considering critical C0 at line 13 instead of line 14 will give us more effective test vectors for the final minimization.

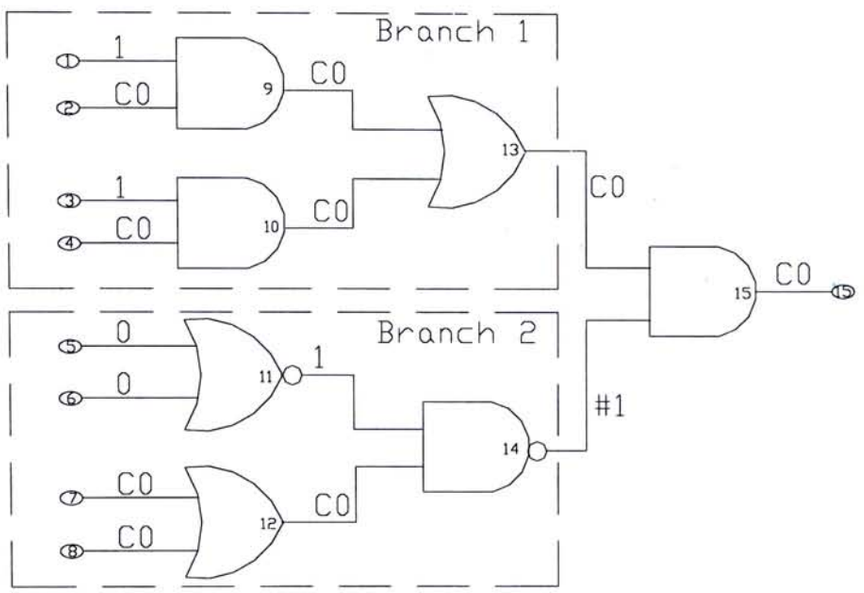


Figure 7.2 Circuit SC2 with line 13 & 14 swapping state

The logic states at lines 1 & 2 can be swapped to generate two more possibilities while lines 3 & 4 can be also be swapped. As a result, there are four more test vectors be put to the Fault Dictionary as follow:

181844448814818-2+5*
=000000440004000-14+3*
=000044000000000-11+2*
811844448814818-2+6*
=000000440004000-14+3*
=000044000000000-11+2*
188144448814818-2+9*
=000000440004000-14+3*
=000044000000000-11+2*
818144448814818-2+10*
=000000440004000-14+3*
=000044000000000-11+2*

Also, we consider the branch 2 of circuit SC2 and lines 11 & 12 can also be swapped as shown in Figure 7.3.

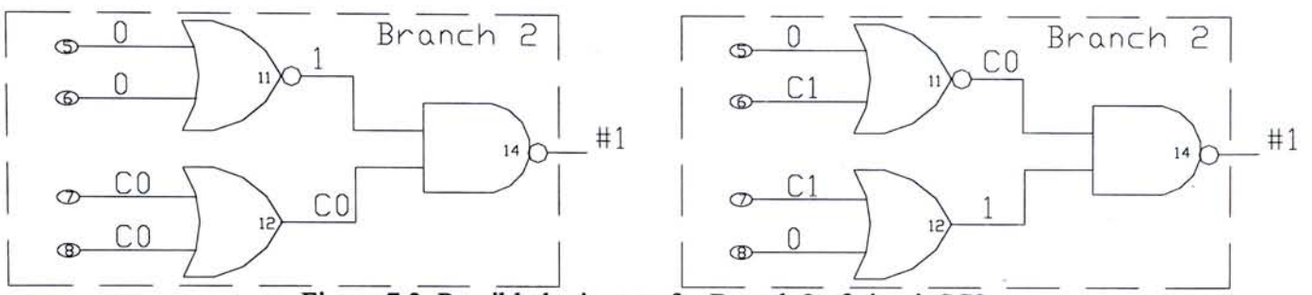


Figure 7.3 Possible logic state for Branch 2 of circuit SC2

By fixing the logic state at branch 1 of circuit SC2, it generates 4 more possible test vectors by swapping lines 5&6 and lines 7&8. If we put both branch 1 and branch 2 into consideration, there are total 4 x 5 = 20 possible test vectors. As a result, there are 28 test vectors being put into the Fault Dictionary by this generation.

Similarly, the second iteration will have 20 test vectors. The third iteration will have 28 test vectors and the last iteration will have 20 test vectors. As a result, a total of 96 test vectors are inside the Fault Dictionary. However, half of them are duplicates and hence only 48 test vectors are listed inside the Fault Dictionary instead of 256 test vectors for minimization.

The advantage of this method can cure the problem of handling reconvergence fanout. For the existing method, signals may crashes at the reconvergence signal line. However, we take those crashes as an alternatives and put both of them into consideration. While they were put into the Fault Dictionary, it can be handled by the out-of-kilter algorithm.

7.3.2 Making the Fault Dictionary by random generation

There are statistical approach for Fault Analysis [30] and random test vectors generation [31,32] to the testability of digital circuits. If we look at the full list of the Fault Dictionary, it consists of many redundant test vectors. Looking at results obtained, it is observed that an effective test vector always tests a lot of stuck-at-faults. Figure 7.4 shows a graph of the number of test vectors versus the number of stuck-at-faults being detected.

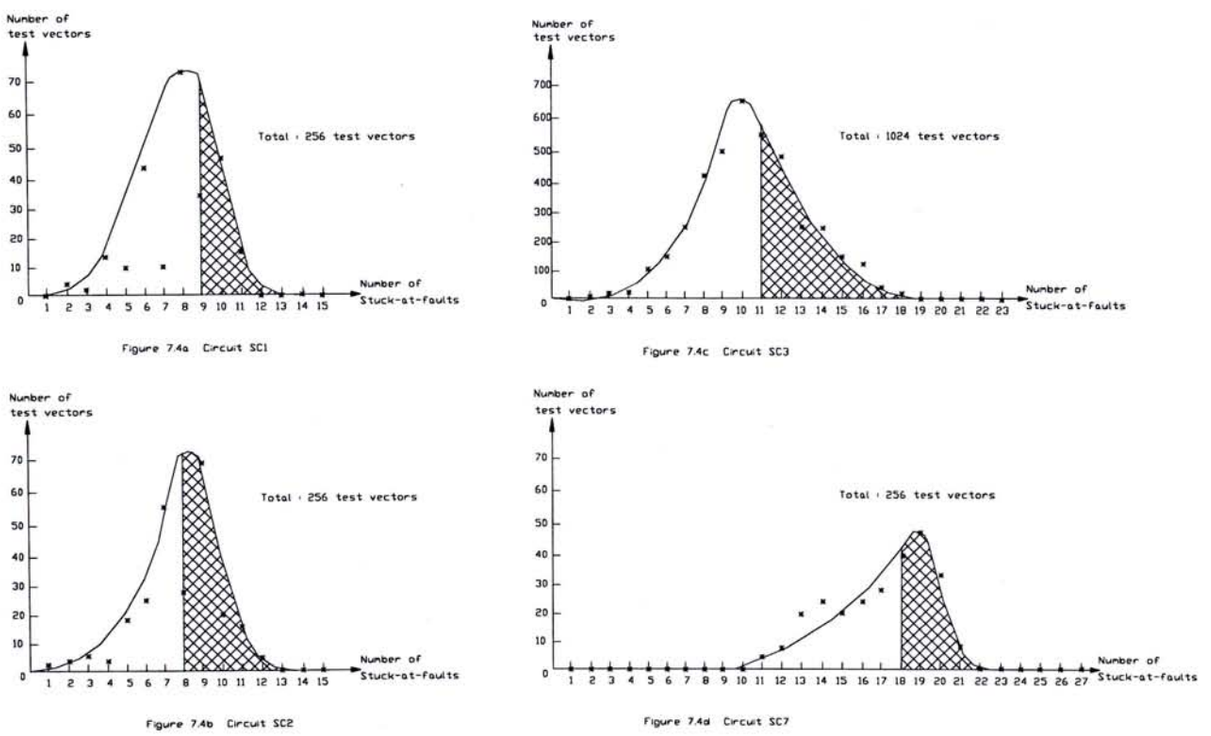


Figure 7.4 No. of test vectors versus No. of stuck at faults being detected

It is observed that the optimum test set obtained by the exhaustive method lies inside the shaded area. And it is obviously simple to delete those test vectors which is outside the shaded area of the graph to make our Fault Dictionary effective. With this deletion, it can save computation time while doing the optimization. The deletion can be done by calling up a test vector from the Fault Dictionary and check for the number of stuck-at-faults. If the number is reasonably large, it is selected and put into the Fault Dictionary. However, there may exist an instance that a critical test vector which detects a specific fault is laid outside the shaded area. If we consider the occurrence of that test vector which tests only one critical fault is extremely low, we can conclude that the possibility of having this instance is very low.

8. Conclusions

8.1 Summary of Results

In high observability testing, internal circuit nodes can be used as test points. Test generation needs to generate a set of test vectors with corresponding internal probings/probe points. For those existing test generation algorithms considering internal probings/probe points, they do not guarantee a true minimum. This thesis offers an optimization method by the out-of-kilter algorithm which will not be affected by fan-out and re-convergence. The main procedure divides into two parts: Part I generates a Fault Dictionary. If we need a true minimum solution, we need to put all possible test vectors and internal probings/probe points to the Fault Dictionary before going to the second part. In Chapter 7.3.1, we suggest to choose only the efficient test vectors generated by the existing test generation algorithm to the Fault Dictionary. Also, we suggest a quick selection of efficient test vectors in Chapter 7.3.2 by considering the number of stuck-at-faults. Part II makes use of the Fault Dictionary and finds the minimum test set.

Table 8.1 summarizes the results obtained by this algorithm on several simple circuits.

Circuits	SC1	SC2	SC3	SC7
No. of s-a-faults	30	30	46	54
No. of test vectors	6	7	9	8
without internal test point	0	0	0	0
Minimum No. of test vectors	4	4	5	4
with optimum No. of probing	4	4	9	4
Minimum No. of test vectors	4	4	5	4
with optimum No. of probe point	2	4	7	2

Table 8.1 Results of the simple circuits by new algorithm

A comparison of the results obtained by this algorithm to those by an existing test generation algorithm [7] is listed in Table 7.1. A significant improvement has been achieved in the new algorithm. Since it considers every possibility and the results can be used as an reference to the other test generation algorithms.

Moreover, the new algorithm can minimize the number of test vectors by fixing the number of internal probings/probe points. The results on a simple logic module is listed in Table 8.2/Table 8.3:

Internal probing	No of test vector	Results	Description
0	8	155551991888480588991188888-2+32 155558848188410588991188888-3+40 155559884188490590959194099-4+48 155551959988440509959194099-4+64 195991555590440905884919909-4+128 199181555509848805848811888-3+188 884811555509148805848811888-4+189 988411555509949005884919909-4+190	No probing
1	7	155551991888480588991188888-2+32 155558848188410588991188888-3+40 155551959988440509959194099-4+64 195991555590440905884919909-4+128 199181555509848805848811888-3+188 884811555509148805848811888-4+189 988005440011959050222913909-1+202 .000005440000050050000000000-23+5	Probe at 23
2	6	155558848188410588991188888-3+40 155551959988440509959194099-4+64 195991555590440905884919909-4+128 884811555509148805848811888-4+189 988005440011959050222913909-1+202 .000005440000050050000000000-23+5 199181551411848844220801828-1+220 .000000550400040044000000000-23+6	Probe at 23 Probe at 23
3	5	155558848188410588991188888-3+40 155551959988440509959194099-4+64 .050550000000000500000000000-22+4 988001551490949044880909909-1+90 .000000550400040044000000000-23+6 884811555509148805848811888-4+189 199185440011858850222811828-1+204 .000005440000050050000000000-23+5	Probe at 22 Probe at 23 Probe at 23
4	4	155149880088494490959090099-2+12 .055040000000404400000000000-22+6 155558848188410588991188888-3+40 .050550000000000500000000000-22+4 988001551490949044880909909-1+90 .000000550400040044000000000-23+6 884811555509148805848811888-4+189 .000000505500000005000000000-23+4	Probe at 22 Probe at 22 Probe at 23 Probe at 23

Table 8.2 Result of circuit sc7 with various number of internal probings

Internal probe point	No of test vector	Results	Description
0	8	155551991888480588991188888-2+32 155558848188410588991188888-3+40 155559884188490590959194099-4+48 155551959988440509959194099-4+64 195991555590440905884919909-4+128 199181555509848805848811888-3+188 884811555509148805848811888-4+189 988411555509949005884919909-4+190	No probe point
1	6	155558848188410588991188888-3+40 155551959988440509959194099-4+64 195991555590440905884919909-4+128 884811555509148805848811888-4+189 988005440011959050222913909-1+202 .000005440000050050000000000-23+5 199181551411848844220801828-1+220 .000000550400040044000000000-23+6	Probe at 23 Probe at 23
2	4	155149880088494490959090099-2+12 .055040000000404400000000000-22+6 155558848188410588991188888-3+40 .050550000000000500000000000-22+4 988001551490949044880909909-1+90 .000000550400040044000000000-23+6 884811555509148805848811888-4+189 .000000505500000005000000000-23+4	Probe at 22 Probe at 22 Probe at 23 Probe at 23

Table 8.3 Result of circuit sc7 with various number of internal probe points

8.2 Further Research

A methodology to obtain a minimum test set with optimum number of internal probings/probe points is developed. However, it is costly and time consuming if large number of primary inputs is being considered. The result obtained can work as an reference for other new test generation algorithms. Since the relationship between the number of additional probings/probe points to the number of internal probings/probe points is difficult to establish, it depends on the characteristic of the circuit and also the weighting factor between having a test vector and having an internal probing/probe point.

However, the algorithm in this thesis already finds the lower bound which is the minimum test set with optimum number of internal probings/probe points and the upper bound which is the conventional method without any internal probings/probe points. In between, to find an efficient method to determine an acceptable solution of a good combination of test vectors with probings/probe points still requires further research.

REFERENCES

1. G. Swan, Y. Trivedi, and D. Wharton, "CrossCheck - A Practical Solution for ASIC Testability," Proc. IEEE International Test Conference, pp.903-908, August 1989.

2. Kerry Pierce, Robert Lipp, Eric Chan and Tony Wong, "High Performance CMOS Array with an Embedded Test Structure," Proceedings of the IEEE 1990 Custom Integrated Circuits Conference, pp.4.1.1-4.1.4, May 1990.

3. Oliver C.S. Choy et al., "Converting a Conventional SEM to an E-Beam Tester," 5th International Symposium on IC Tecchnology, Systems & Applications, 1993.

4. Teruo Tamama and Norio Kuji, "Integrating an Electron-Beam System into VLSI Fault Diagnosis," IEEE Design and Test, pp.23-29, August 1986.

5. D.W. Ranasinghe, "Advances in Electron Beam Testing," Testing and Diagnosis of VLSI and ULSI, Kluwer Academic Publishers, pp. 509-526, 1988.

6. J. Mucha, "Bridging the Gap between Conventional and E-Beam Testing," Microelectronic Engineering, Vol. 16, pp.185-192, Special issue on Electron and Optical Beam Testing of Integrated Circuits.

7. Oliver C.S. Choy, L.K. Chan, Ray Chan and C.F. Chan, "Test Generation for E-beam Testing of VLSI Circuits," IEEE Proceeding of the Second Asian Test Symposium, pp. 101- 106, Nov 1993.

8. Kinch, Richard John, "Automatic Test Generation for Electron-beam Testing of VLSI Circuits," Ph.D. thesis, School of Electrical Engineering, Cornell University, Ithaca, NY14853, May 1982.

9. D.T. Wang, "An Algorithm for the Generation of Test Sets for Combinational Logic Networks," IEEE Transactions on Computers, Vol.C-24, No.7, pp.742-756, July 1975.

10. J.W. Lamont and D.J. Lubash, "Daily Power System Fuel Optimization using a Network Flow Algorithm," Proceedings of the 21th Annual North America Power Symposium, pp.12-20, Oct 1989.

11. J.Z. Zhu and G.Y. Xu, "Approach to automatic contingency selection by reactive type performance index," IEEE Proc. [Generation, Transmission and Distribution],"
Vol.138, Iss 1, pp.65-68, Jan 1991.

12. Bruneo Codenotti and Roberto Tamassia, "A Network Flow Approach to the Reconfiguration of VLSI Arrays," IEEE Transactions on Computers, Vol.40, Iss 1, pp.118-121, Jan 1991.

13. D.T. Philips and P.A. Jensen, "The out-of-kilter algorithm," Industrial Engineering, pp.36-44, 1974.

14. B.L. Redmond, "Scanning Electron Microscope Operators Manual AMRAY Models 1820, 1830," AMRAY Inc. 1990.

15. M. Macari, K. Thangamuthu and S. Cohen, "Automated Contactless SEM Testing for VLSI Development and Failure Analysis," IEEE Proceedings of IRPS, pp.163-166, 1982.

16. Kirk J. Bertsche and Harry K. Charles, "The Practical Implementation of Voltage Contrast as a Diagnostic Tools," IEEE Proceedings of IRPS, pp.167-178, 1982.

17. C. Morandi, M. Vanzi, F.Bianco and R. Neri, "a PC-AT-Based System for the Acquisition of SEM Images," Scanning Vol.11, pp.81-85, 1989.

18. Norio Kuji, Teruo Tamama and Takao Yano, "A Fully-Automated Electron Beam Test System for VLSI Circuits," IEEE Design and Test, pp.74-82, Oct 1982.

19. J.C.H. Phang, K.K.W. Ong, D.S.H. Chan and T.S. Low, "SEM Image Collection and Processing System (SEMICAPS) Part I-Collection System," Proc. Int. Symp. Physical and Failure Analysis of Integrated Circuits, Singapore, pp.50-55, Oct 1987.

20. Oliver C.S. Choy, R. Chan and C.F. Chan, "Conventional SEM for Digital Logic Verification," 5th International Symposium on IC Technology, System and Applications, pp.737-740, Sept. 1993.

21. I.S.M. Chin, W.K. Chim, P.C.T. Koh, D.S.H. Chan, T.S. Low, J.C.H. Phang and P.Y.S. Ho, "Observation of IC surface Potentials with Voltage Contrast," Proc. NUS Symp. Applications of Scanning Microscopy, pp.82-87, Sept. 1988.

22. Abramovici, P.R. Menon and D.T. Miller, "Critical Path Tracing: An Alternative to Fault Simulation," IEEE Design and Test of Computers, Vol.1, No.1, pp.83-93, Feb 1984.

23. J.P. Roth, W.G. Bouricius and P.R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," IEEE Trans. on Electronic Computers, Vol. EC-16, No.10, pp.567-579, Oct 1967.

24. Fadi Maamari and Janusz Rajski, "Reconvergent Fanout Analysis and Fault Simulation Complexity of Combinational Circuits," Electronic Letter, Vol.23, No.21, pp.1131-1133, Oct 1987.

25. Menon P., Levendel Y. and Abramovici M., "Critical Path Tracing in Sequential Circuit," Proceedings of International Conference on Computer-Aided Design, pp. 162-165, 1988.

26. Tomasz Radzik, "Faster algorithms for the Generalized Network Flow Problem," Proceedings 34th Annual Symposium on Foundation of Computer Science, pp. 438-448, 1993.

27. Don. T. Philips and Alberto Garcia-Diaz, "Fundamentals of Network Analysis," Prentice Hall International Series in Industrial and Systems Engineering, pp. 204-267.

28. Hamdy A. Taha, "Operations Research", Macmillan Publishing Company, pp. 115-170, 1982.

29. Kinch R. And Pottle C., "Automatic Test Generation for Electron-beam Testing of VLSI circuits," Proceedings IEEE ICCC, pp. 548-551, 1982.

30. M.N.R.D. Yacoub and P.D.Noakes, "An Investigation of Statistical Fault Analysis for Combinational Logic Circuits," IEE Colloquium on 'Design for Testability', Vol. 32, pp. 5/1-4, March 1988.

31. S.C. Seth, V.D. Agrawal and H. Farhat, "A Theory of Testability with Application to Fault Coverage Analysis," Proceeding of the 1st European Test Conference, pp. 139-143, April 1989.

32. W.H. Debany, C.R.P. Hartmann, P.K. Varshney and K.G. Mehrotra, "Comparison of Random Test Vector Generation Strategies," IEEE International Conference on Computer Aided Design, pp. 244-247, Nov 1991.

Appendix A

000044442200002-2+0*
.000000440000000-12+2*
.000044000000000-11+2*
140044444200002-3+1*
.040000004000000-13+2*
.000000440000000-12+2*
.000044000000000-11+2*
410044444200002-3+2*
.400000004000000-13+2*
.000000440000000-12+2*
.000044000000000-11+2*
550044445000122-3+3*
.550000005000000-13+3*
.000000440000000-12+2*
.000044000000000-11+2*
001444442400002-3+4*
.000400000400000-13+2*
.000000440000000-12+2*
.000044000000000-11+2*
141444444400002-3+5*
.040400004400000-13+4*
.000000440000000-12+2*
.000044000000000-11+2*
411444444400002-3+6*
.400400004400000-13+4*
.000000440000000-12+2*
.000044000000000-11+2*
551444445000122-4+7*
.550000005000000-13+3*
.000000440000000-12+2*
.000044000000000-11+2*
.000400000000000-10+1*
004144442400002-3+8*
.004000000400000-13+2*
.000000440000000-12+2*
.000044000000000-11+2*
144144444400002-3+9*
.044000004400000-13+4*
.000000440000000-12+2*
.000044000000000-11+2*
414144444400002-3+10*
.404000004400000-13+4*
.000000440000000-12+2*
.000044000000000-11+2*
554144445000122-4+11*
.550000005000000-13+3*
.000000440000000-12+2*
.000044000000000-11+2*
.004000000000000-10+1*
005544440500122-3+12*
.005500000500000-13+3*
.000000440000000-12+2*
.000044000000000-11+2*
145544440500122-4+13*
.005500000500000-13+3*
.000000440000000-12+2*
.000044000000000-11+2*
.040000000000000-9+1*
415544440500122-4+14*
.005500000500000-13+3*

.000000440000000-12+2*
.000044000000000-11+2*
.400000000000000-9+1*
555544441100122-4+15*
.000000440000000-12+2*
.000044000000000-11+2*
.005500000000000-10+2*
.550000000000000-9+2*
000050442214002-2+16*
.000000440004000-14+3*
.000050000000000-11+1*
140050444214002-3+17*
.000000440004000-14+3*
.040000004000000-13+2*
.000050000000000-11+1*
410050444214002-3+18*
.000000440004000-14+3*
.400000004000000-13+2*
.000050000000000-11+1*
550050885018188-2+19*
.550000005000000-13+3*
.000050000000000-11+1*
001450442414002-3+20*
.000000440004000-14+3*
.000400000400000-13+2*
.000050000000000-11+1*
141450444414002-3+21*
.000000440004000-14+3*
.040400004400000-13+4*
.000050000000000-11+1*
411450444414002-3+22*
.000000440004000-14+3*
.400400004400000-13+4*
.000050000000000-11+1*
551450885018188-3+23*
.550000005000000-13+3*
.000050000000000-11+1*
.000400000000000-10+1*
004150442414002-3+24*
.000000440004000-14+3*
.004000000400000-13+2*
.000050000000000-11+1*
144150444414002-3+25*
.000000440004000-14+3*
.044000004400000-13+4*
.000050000000000-11+1*
414150444414002-3+26*
.000000440004000-14+3*
.404000004400000-13+4*
.000050000000000-11+1*
554150885018188-3+27*
.550000005000000-13+3*
.000050000000000-11+1*
.004000000000000-10+1*
005550880518188-2+28*
.005500000500000-13+3*
.000050000000000-11+1*
145550880518188-3+29*
.005500000500000-13+3*
.000050000000000-11+1*

.040000000000000-9+1*
415550880518188-3+30*
.005500000500000-13+3*
.000050000000000-11+1*
.400000000000000-9+1*
555550881118188-3+31*
.000050000000000-11+1*
.005500000000000-10+2*
.550000000000000-9+2*
000005442214002-2+32*
.000000440004000-14+3*
.000005000000000-11+1*
140005444214002-3+33*
.000000440004000-14+3*
.040000004000000-13+2*
.000005000000000-11+1*
410005444214002-3+34*
.000000440004000-14+3*
.400000004000000-13+2*
.000005000000000-11+1*
550005885018188-2+35*
.550000005000000-13+3*
.000005000000000-11+1*
001405442414002-3+36*
.000000440004000-14+3*
.000400000400000-13+2*
.000005000000000-11+1*
141405444414002-3+37*
.000000440004000-14+3*
.040400004400000-13+4*
.000005000000000-11+1*
411405444414002-3+38*
.000000440004000-14+3*
.400400004400000-13+4*
.000005000000000-11+1*
551405885018188-3+39*
.550000005000000-13+3*
.000005000000000-11+1*
.000400000000000-10+1*
004105442414002-3+40*
.000000440004000-14+3*
.004000000400000-13+2*
.000005000000000-11+1*
144105444414002-3+41*
.000000440004000-14+3*
.044000004400000-13+4*
.000005000000000-11+1*
414105444414002-3+42*
.000000440004000-14+3*
.404000004400000-13+4*
.000005000000000-11+1*
554105885018188-3+43*
.550000005000000-13+3*
.000005000000000-11+1*
.004000000000000-10+1*
005505880518188-2+44*
.005500000500000-13+3*
.000005000000000-11+1*
145505880518188-3+45*
.005500000500000-13+3*

.000005000000000-11+1*
.040000000000000-9+1*
415505880518188-3+46*
.005500000500000-13+3*
.000005000000000-11+1*
.400000000000000-9+1*
555505881118188-3+47*
.000005000000000-11+1*
.005500000000000-10+2*
.550000000000000-9+2*
000011442214002-1+48*
.000000440004000-14+3*
140011444214002-2+49*
.000000440004000-14+3*
.040000004000000-13+2*
410011444214002-2+50*
.000000440004000-14+3*
.400000004000000-13+2*
550011885018188-1+51*
.550000005000000-13+3*
001411442414002-2+52*
.000000440004000-14+3*
.000400000400000-13+2*
141411444414002-2+53*
.000000440004000-14+3*
.040400004400000-13+4*
411411444414002-2+54*
.000000440004000-14+3*
.400400004400000-13+4*
551411885018188-2+55*
.550000005000000-13+3*
.000400000000000-10+1*
004111442414002-2+56*
.000000440004000-14+3*
.004000000400000-13+2*
144111444414002-2+57*
.000000440004000-14+3*
.044000004400000-13+4*
4141111444414002-2+58*
.000000440004000-14+3*
.404000004400000-13+4*
554111885018188-2+59*
.550000005000000-13+3*
.004000000000000-10+1*
005511880518188-1+60*
.005500000500000-13+3*
145511880518188-2+61*
.005500000500000-13+3*
.040000000000000-9+1*
415511880518188-2+62*
.005500000500000-13+3*
.400000000000000-9+1*
555511881118188-2+63*
.005500000000000-10+2*
.550000000000000-9+2*
000044502241002-2+64*
.000044000040000-14+3*
.000000500000000-12+1*

140044504241002-3+65*
.000044000040000-14+3*
.040000004000000-13+2*
.000000500000000-12+1*
410044504241002-3+66*
.000044000040000-14+3*
.400000004000000-13+2*
.000000500000000-12+1*
550088505081188-2+67*
.550000005000000-13+3*
.000000500000000-12+1*
001444502441002-3+68*
.000044000040000-14+3*
.000400000400000-13+2*
.000000500000000-12+1*
141444504441002-3+69*
.000044000040000-14+3*
.040400004400000-13+4*
.000000500000000-12+1*
411444504441002-3+70*
.000044000040000-14+3*
.400400004400000-13+4*
.000000500000000-12+1*
551488505081188-3+71*
.550000005000000-13+3*
.000000500000000-12+1*
.000400000000000-10+1*
004144502441002-3+72*
.000044000040000-14+3*
.004000000400000-13+2*
.000000500000000-12+1*
144144504441002-3+73*
.000044000040000-14+3*
.044000004400000-13+4*
.000000500000000-12+1*
414144504441002-3+74*
.000044000040000-14+3*
.404000004400000-13+4*
.000000500000000-12+1*
554188505081188-3+75*
.550000005000000-13+3*
.000000500000000-12+1*
.004000000000000-10+1*
005588500581188-2+76*
.005500000500000-13+3*
.000000500000000-12+1*
145588500581188-3+77*
.005500000500000-13+3*
.000000500000000-12+1*
.040000000000000-9+1*
415588500581188-3+78*
.005500000500000-13+3*
.000000500000000-12+1*
.400000000000000-9+1*
555588501181188-3+79*
.000000500000000-12+1*
.005500000000000-10+2*
.550000000000000-9+2*

000050502255212-1+80*
.000050500055000-14+4*
180050508255818-1+81*
.000050500055000-14+4*
810050508255818-1+82*
.000050500055000-14+4*
990090909099999-0+83*
001850502855818-1+84*
.000050500055000-14+4*
181850508855818-1+85*
.000050500055000-14+4*
811850508855818-1+86*
.000050500055000-14+4*
991490909099999-1+87*
.000400000000000-10+1*
008150502855818-1+88*
.000050500055000-14+4*
188150508855818-1+89*
.000050500055000-14+4*
818150508855818-1+90*
.000050500055000-14+4*
994190909099999-1+91*
.004000000000000-10+1*
009990900999999-0+92*
149990900999999-1+93*
.040000000000000-9+1*
419990900999999-1+94*
.400000000000000-9+1*
555590901199399-2+95*
.005500000000000-10+2*
.550000000000000-9+2*
000005502255212-1+96*
.000005500055000-14+4*
180005508255818-1+97*
.000005500055000-14+4*
810005508255818-1+98*
.000005500055000-14+4*
990009909099999-0+99*
001805502855818-1+100*
.000005500055000-14+4*
181805508855818-1+101*
.000005500055000-14+4*
811805508855818-1+102*
.000005500055000-14+4*
991409909099999-1+103*
.000400000000000-10+1*
008105502855818-1+104*
.000005500055000-14+4*
188105508855818-1+105*
.000005500055000-14+4*
818105508855818-1+106*
.000005500055000-14+4*
994109909099999-1+107*
.004000000000000-10+1*
009909900999999-0+108*
149909900999999-1+109*
.040000000000000-9+1*
419909900999999-1+110*

.400000000000000-9+1*
555509901199399-2+111*
.005500000000000-10+2*
.550000000000000-9+2*
000011502235212-1+112*
.000000500005000-14+2*
180011508235818-1+113*
.000000500005000-14+2*
810011508235818-1+114*
.000000500005000-14+2*
990011909039999-0+115*
001811502835818-1+116*
.000000500005000-14+2*
181811508835818-1+117*
.000000500005000-14+2*
811811508835818-1+118*
.000000500005000-14+2*
991411909039999-1+119*
.000400000000000-10+1*
008111502835818-1+120*
.000000500005000-14+2*
188111508835818-1+121*
.000000500005000-14+2*
818111508835818-1+122*
.000000500005000-14+2*
994111909039999-1+123*
.004000000000000-10+1*
009911900939999-0+124*
149911900939999-1+125*
.040000000000000-9+1*
419911900939999-1+126*
.400000000000000-9+1*
555511901139399-2+127*
.005500000000000-10+2*
.550000000000000-9+2*
000044052241002-2+128*
.000044000040000-14+3*
.000000050000000-12+1*
140044054241002-3+129*
.000044000040000-14+3*
.040000004000000-13+2*
.000000050000000-12+1*
410044054241002-3+130*
.000044000040000-14+3*
.400000004000000-13+2*
.000000050000000-12+1*
550088055081188-2+131*
.550000005000000-13+3*
.000000050000000-12+1*
001444052441002-3+132*
.000044000040000-14+3*
.000400000400000-13+2*
.000000050000000-12+1*
141444054441002-3+133*
.000044000040000-14+3*
.040400004400000-13+4*
.000000050000000-12+1*
411444054441002-3+134*

.000044000040000-14+3*
.400400004400000-13+4*
.000000050000000-12+1*
551488055081188-3+135*
.550000005000000-13+3*
.000000050000000-12+1*
.000400000000000-10+1*
004144052441002-3+136*
.000044000040000-14+3*
.004000000400000-13+2*
.000000050000000-12+1*
144144054441002-3+137*
.000044000040000-14+3*
.044000004400000-13+4*
.000000050000000-12+1*
414144054441002-3+138*
.000044000040000-14+3*
.404000004400000-13+4*
.000000050000000-12+1*
554188055081188-3+139*
.550000005000000-13+3*
.000000050000000-12+1*
.004000000000000-10+1*
005588050581188-2+140*
.005500000500000-13+3*
.000000050000000-12+1*
145588050581188-3+141*
.005500000500000-13+3*
.000000050000000-12+1*
.040000000000000-9+1*
415588050581188-3+142*
.005500000500000-13+3*
.000000050000000-12+1*
.400000000000000-9+1*
555588051181188-3+143*
.000000050000000-12+1*
.005500000000000-10+2*
.550000000000000-9+2*
000050052255212-1+144*
.000050050055000-14+4*
180050058255818-1+145*
.000050050055000-14+4*
810050058255818-1+146*
.000050050055000-14+4*
990090099099999-0+147*
001850052855818-1+148*
.000050050055000-14+4*
181850058855818-1+149*
.000050050055000-14+4*
811850058855818-1+150*
.000050050055000-14+4*
991490099099999-1+151*
.000400000000000-10+1*
008150052855818-1+152*
.000050050055000-14+4*
188150058855818-1+153*
.000050050055000-14+4*
818150058855818-1+154*

.000050050055000-14+4*
994190099099999-1+155*
.004000000000000-10+1*
009990090999999-0+156*
149990090999999-1+157*
.040000000000000-9+1*
419990090999999-1+158*
.400000000000000-9+1*
555590091199399-2+159*
.005500000000000-10+2*
.550000000000000-9+2*
000005052255212-1+160*
.000005050055000-14+4*
180005058255818-1+161*
.000005050055000-14+4*
810005058255818-1+162*
.000005050055000-14+4*
990009099099999-0+163*
001805052855818-1+164*
.000005050055000-14+4*
181805058855818-1+165*
.000005050055000-14+4*
811805058855818-1+166*
.000005050055000-14+4*
991409099099999-1+167*
.000400000000000-10+1*
008105052855818-1+168*
.000005050055000-14+4*
188105058855818-1+169*
.000005050055000-14+4*
818105058855818-1+170*
.000005050055000-14+4*
994109099099999-1+171*
.004000000000000-10+1*
009909090999999-0+172*
149909090999999-1+173*
.040000000000000-9+1*
419909090999999-1+174*
.400000000000000-9+1*
555509091199399-2+175*
.005500000000000-10+2*
.550000000000000-9+2*
000011052235212-1+176*
.000000050005000-14+2*
180011058235818-1+177*
.000000050005000-14+2*
810011058235818-1+178*
.000000050005000-14+2*
990011099039999-0+179*
001811052835818-1+180*
.000000050005000-14+2*
181811058835818-1+181*
.000000050005000-14+2*
811811058835818-1+182*
.000000050005000-14+2*
991411099039999-1+183*
.000400000000000-10+1*
008111052835818-1+184*

.000000050005000-14+2*
188111058835818-1+185*
.000000050005000-14+2*
818111058835818-1+186*
.000000050005000-14+2*
994111099039999-1+187*
.004000000000000-10+1*
009911090939999-0+188*
149911090939999-1+189*
.040000000000000-9+1*
419911090939999-1+190*
.400000000000000-9+1*
555511091139399-2+191*
.005500000000000-10+2*
.550000000000000-9+2*
000044112241002-1+192*
.000044000040000-14+3*
140044114241002-2+193*
.000044000040000-14+3*
.040000004000000-13+2*
410044114241002-2+194*
.000044000040000-14+3*
.400000004000000-13+2*
550088115081188-1+195*
.550000005000000-13+3*
001444112441002-2+196*
.000044000040000-14+3*
.000400000400000-13+2*
141444114441002-2+197*
.000044000040000-14+3*
.040400004400000-13+4*
411444114441002-2+198*
.000044000040000-14+3*
.400400004400000-13+4*
551488115081188-2+199*
.550000005000000-13+3*
.000400000000000-10+1*
004144112441002-2+200*
.000044000040000-14+3*
.004000000400000-13+2*
144144114441002-2+201*
.000044000040000-14+3*
.044000004400000-13+4*
414144114441002-2+202*
.000044000040000-14+3*
.404000004400000-13+4*
554188115081188-2+203*
.550000005000000-13+3*
.004000000000000-10+1*
005588110581188-1+204*
.005500000500000-13+3*
145588110581188-2+205*
.005500000500000-13+3*
.040000000000000-9+1*
415588110581188-2+206*
.005500000500000-13+3*
.400000000000000-9+1*
555588111181188-2+207*
.005500000000000-10+2*
.550000000000000-9+2*
000050112253212-1+208*
.000050000050000-14+2*
180050118253818-1+209*
.000050000050000-14+2*
810050118253818-1+210*
.000050000050000-14+2*
990090119093999-0+211*
001850112853818-1+212*
.000050000050000-14+2*
181850118853818-1+213*
.000050000050000-14+2*
811850118853818-1+214*
.000050000050000-14+2*
991490119093999-1+215*
.000400000000000-10+1*
008150112853818-1+216*
.000050000050000-14+2*
188150118853818-1+217*
.000050000050000-14+2*
818150118853818-1+218*
.000050000050000-14+2*
994190119093999-1+219*
.004000000000000-10+1*
009990110993999-0+220*
149990110993999-1+221*
.040000000000000-9+1*
419990110993999-1+222*
.400000000000000-9+1*
555590111193399-2+223*
.005500000000000-10+2*
.550000000000000-9+2*
000005112253212-1+224*
.000005000050000-14+2*
180005118253818-1+225*
.000005000050000-14+2*
810005118253818-1+226*
.000005000050000-14+2*
990009119093999-0+227*
001805112853818-1+228*
.000005000050000-14+2*
181805118853818-1+229*
.000005000050000-14+2*
811805118853818-1+230*
.000005000050000-14+2*
991409119093999-1+231*
.000400000000000-10+1*
008105112853818-1+232*
.000005000050000-14+2*
188105118853818-1+233*
.000005000050000-14+2*
818105118853818-1+234*
.000005000050000-14+2*
994109119093999-1+235*
.004000000000000-10+1*
009909110993999-0+236*
149909110993999-1+237*
419909110993999-1+238*
.400000000000000-9+1*
555509111193399-2+239*
.005500000000000-10+2*
.550000000000000-9+2*
000011112233212-0+240*
180011118233818-0+241*
810011118233818-0+242*
990011119033939-0+243*
001811112833818-0+244*
181811118833818-0+245*
811811118833818-0+246*
991411119033939-1+247*
.000400000000000-10+1*
008111112833818-0+248*
188111118833818-0+249*
818111118833818-0+250*
994111119033939-1+251*
.004000000000000-10+1*
009911110933939-0+252*
149911110933939-1+253*
.040000000000000-9+1*
419911110933939-1+254*
.400000000000000-9+1*
555511111133333-2+255*
.005500000000000-10+2*
.550000000000000-9+2*
End

Appendix B

444008440044114282551080288-4+0*
.400000000000004000000000000-22+2*
.000000000440000000550000000-24+4*
.000000440000000000000000000-14+2*
.044000000000000000000000000-13+2*
544008440088515082991188888-2+1*
.544000000000505000000000000-22+5*
.000000440000000000000000000-14+2*
055148440044012482551080288-4+2*
.000040000000000400000000000-22+2*
.000000000044000000550000000-24+4*
.000000440000000000000000000-14+2*
.055000000000000000000000000-13+2*
155148440044414482551080288-3+3*
.055040000000404400000000000-22+6*
.000000000044000000550000000-24+4*
.000000440000000000000000000-14+2*
444418440044114482551080288-4+4*
.440400000000004400000000000-22+5*
.000000000044000000550000000-24+4*
.000000440000000000000000000-14+2*
.044000000000000000000000000-13+2*
544418440088515082991188888-3+5*
.544000000000505000000000000-22+5*
.040400000000000000000000000-16+2*
.000000440000000000000000000-14+2*
055558440088010582991188888-3+6*
.050550000000000500000000000-22+4*
.000000440000000000000000000-14+2*
.055000000000000000000000000-13+2*
155558440088410582991188888-3+7*
.050550000000000500000000000-22+4*
.055000000000400000000000000-15+3*
.000000440000000000000000000-14+2*
444009880088194290959090099-3+8*
.400000000000004000000000000-22+2*
.000000000000000000050000000-24+1*
.044000000000000000000000000-13+2*
544009880088595090959194099-2+9*
.000000000000000000050004000-25+2*
.544000000000505000000000000-22+5*
055149880088092490959090099-3+10*
.000040000000000400000000000-22+2*
.000000000000000000050000000-24+1*
.055000000000000000000000000-13+2*
155149880088494490959090099-2+11*
.055040000000404400000000000-22+6*
.000000000000000000050000000-24+1*
444419880088194490959090099-3+12*
.440400000000004400000000000-22+5*
.000000000000000000050000000-24+1*
.044000000000000000000000000-13+2*
544419880088595090959194099-3+13*
.000000000000000000050004000-25+2*
.544000000000505000000000000-22+5*
.040400000000000000000000000-16+2*
055559880088090590959194099-3+14*
.000000000000000000050004000-25+2*
.050550000000000500000000000-22+4*
.055000000000000000000000000-13+2*

155559880088490590959194099-3+15*
.000000000000000000050004000-25+2*
.050550000000000500000000000-22+4*
.055000000000400000000000000-15+3*
444000551844104228551080288-4+16*
.400000000000004000000000000-22+2*
.000000000044000000550000000-24+4*
.000000550000000000000000000-14+2*
.044000000000000000000000000-13+2*
544000551888505028991188888-2+17*
.544000000000505000000000000-22+5*
.000000550000000000000000000-14+2*
055140551844002428551080288-4+18*
.000040000000000400000000000-22+2*
.000000000044000000550000000-24+4*
.000000550000000000000000000-14+2*
.055000000000000000000000000-13+2*
155140551844404428551080288-3+19*
.055040000000404400000000000-22+6*
.000000000044000000550000000-24+4*
.000000550000000000000000000-14+2*
444410551844104428551080288-4+20*
.440400000000004400000000000-22+5*
.000000000044000000550000000-24+4*
.000000550000000000000000000-14+2*
.044000000000000000000000000-13+2*
544410551888505028991188888-3+21*
.544000000000505000000000000-22+5*
.040400000000000000000000000-16+2*
.000000550000000000000000000-14+2*
055550551888000528991188888-3+22*
.050550000000000500000000000-22+4*
.000000550000000000000000000-14+2*
.055000000000000000000000000-13+2*
155550551888400528991188888-3+23*
.050550000000000500000000000-22+4*
.055000000000400000000000000-15+3*
.000000550000000000000000000-14+2*
444001991844184288551080288-3+24*
.400000000000004000000000000-22+2*
.000000000044000000550000000-24+4*
.044000000000000000000000000-13+2*
544001991888585088991188888-1+25*
.544000000000505000000000000-22+5*
055141991844082488551080288-3+26*
.000040000000000400000000000-22+2*
.000000000044000000550000000-24+4*
.055000000000000000000000000-13+2*
155141991844484488551080288-2+27*
.055040000000404400000000000-22+6*
.000000000044000000550000000-24+4*
444411991844184488551080288-3+28*
.440400000000004400000000000-22+5*
.000000000044000000550000000-24+4*
.044000000000000000000000000-13+2*
544411991888585088991188888-2+29*
.544000000000505000000000000-22+5*
.040400000000000000000000000-16+2*
055551991888080588991188888-2+30*
.050550000000000500000000000-22+4*

.055000000000000000000000000-13+2*
15555199188848058899118888-2+31*
.050550000000000500000000000-22+4*
.055000000000400000000000000-15+3*
44400884814411428855108028-4+32*
.400000000000004000000000000-22+2*
.000000000044000000550000000-24+4*
.000000040000000000000000000-14+1*
.044000000000000000000000000-13+2*
54400884818851508899118888-2+33*
.544000000000505000000000000-22+5*
.000000040000000000000000000-14+1*
05514884814401248855108028-4+34*
.000040000000000400000000000-22+2*
.000000000044000000550000000-24+4*
.000000040000000000000000000-14+1*
.055000000000000000000000000-13+2*
15514884814441448855108028-3+35*
.055040000000404400000000000-22+6*
.000000000044000000550000000-24+4*
.000000040000000000000000000-14+1*
44441884814411448855108028-4+36*
.440400000000004400000000000-22+5*
.000000000044000000550000000-24+4*
.000000040000000000000000000-14+1*
.044000000000000000000000000-13+2*
54441884818851508899118888-3+37*
.544000000000505000000000000-22+5*
.040400000000000000000000000-16+2*
.000000040000000000000000000-14+1*
05555884818801058899118888-3+38*
.050550000000000500000000000-22+4*
.000000040000000000000000000-14+1*
.055000000000000000000000000-13+2*
15555884818841058899118888-3+39*
.050550000000000500000000000-22+4*
.055000000000400000000000000-15+3*
.000000040000000000000000000-14+1*
44400988418819429095909099-4+40*
.400000000000004000000000000-22+2*
.000000000000000000050000000-24+1*
.000000040000000000000000000-18+1*
.044000000000000000000000000-13+2*
54400988418859509095919409 9-3+41*
.000000000000000000050004000-25+2*
.544000000000505000000000000-22+5*
.000000040000000000000000000-18+1*
05514988418809249095909099-4+42*
.000040000000000400000000000-22+2*
.000000000000000000050000000-24+1*
.000000040000000000000000000-18+1*
.055000000000000000000000000-13+2*
15514988418849449095909099-3+43*
.055040000000404400000000000-22+6*
.000000000000000000050000000-24+1*
.000000040000000000000000000-18+1*
44441988418819449095909099-4+44*
.440400000000004400000000000-22+5*
.000000000000000000050000000-24+1*
.000000040000000000000000000-18+1*
.044000000000000000000000000-13+2*
54441095998850500995919409 9-4+53*
.000000000000000000050004000-25+2*
.544000000000505000000000000-22+5*
.040400000000000000000000000-16+2*
.000000050000000000000000000-14+1*
05555095998800050995919409 9-4+54*
.000000000000000000050004000-25+2*
.050550000000000500000000000-22+4*
.000000050000000000000000000-14+1*
.055000000000000000000000000-13+2*
15555095998840050995919409 9-4+55*
.000000000000000000050004000-25+2*
.050550000000000500000000000-22+4*
.055000000000400000000000000-15+3*
.000000050000000000000000000-14+1*
44400195998814420995909099-4+56*
.400000000000004000000000000-22+2*

.000000000000000000050000000-24+1*
.000000050000040000000000000-17+2*
.044000000000000000000000000-13+2*
5440019599885450099591940 99-3+57*
.000000000000000000050004000-25+2*
.544000000000505000000000000-22+5*
.000000050000040000000000000-17+2*
055141959988042409959090099-4+58*
.000040000000000400000000000-22+2*
.000000000000000000050000000-24+1*
.000000050000040000000000000-17+2*
.055000000000000000000000000-13+2*
155141959988444409959090099-3+59*
.055040000000404400000000000-22+6*
.000000000000000000050000000-24+1*
.000000050000040000000000000-17+2*
444411959988144409959090099-4+60*
.440400000000004400000000000-22+5*
.000000000000000000050000000-24+1*
.000000050000040000000000000-17+2*
.044000000000000000000000000-13+2*
5444119599885450099591940 99-4+61*
.000000000000000000050004000-25+2*
.544000000000505000000000000-22+5*
.000000050000040000000000000-17+2*
.040400000000000000000000000-16+2*
055551959988040509959194099-4+62*
.000000000000000000050004000-25+2*
.050550000000000500000000000-22+4*
.000000050000040000000000000-17+2*
.055000000000000000000000000-13+2*
155551959988440509959194099-4+63*
.000000000000000000050004000-25+2*
.050550000000000500000000000-22+4*
.000000050000040000000000000-17+2*
.055000000000400000000000000-15+3*
844004440050118242440801828-4+64*
.000004000000000040000000000-23+2*
.000000000050000000440000000-24+3*
.000000440000000000000000000-14+2*
.044000000000000000000000000-13+2*
988004440090919042880909909-2+65*
.000004000000000040000000000-23+2*
.000000440000000000000000000-14+2*
055184440050012842440801828-4+66*
.000004000000000040000000000-23+2*
.000000000050000000440000000-24+3*
.000000440000000000000000000-14+2*
.055000000000000000000000000-13+2*
199184440050818842440801828-3+67*
.000004000000000040000000000-23+2*
.000000000050000000440000000-24+3*
.000000440000000000000000000-14+2*
884814440050118842440801828-4+68*
.000004000000000040000000000-23+2*
.000000000050000000440000000-24+3*
.000000440000000000000000000-14+2*
.004000000000000000000000000-13+1*

988414440090919042880909909-3+69*
.000004000000000040000000000-23+2*
.000400000000000000000000000-16+1*
.000000440000000000000000000-14+2*
095994440090010942880909909-3+70*
.000004000000000040000000000-23+2*
.000000440000000000000000000-14+2*
.005000000000000000000000000-13+1*
195994440090410942880909909-3+71*
.000004000000000040000000000-23+2*
.005000000000400000000000000-15+2*
.000000440000000000000000000-14+2*
844005440090158250848811888-3+72*
.000005440000050050000000000-23+5*
.000000000000000000040000000-24+1*
.044000000000000000000000000-13+2*
988005440090959050884919909-2+73*
.000000000000000000004000000-26+1*
.000005440000050050000000000-23+5*
055185440090052850848811888-3+74*
.000005440000050050000000000-23+5*
.000000000000000000040000000-24+1*
.055000000000000000000000000-13+2*
199185440090858850848811888-2+75*
.000005440000050050000000000-23+5*
.000000000000000000040000000-24+1*
884815440090158850848811888-3+76*
.000005440000050050000000000-23+5*
.000000000000000000040000000-24+1*
.004000000000000000000000000-13+1*
988415440090959050884919909-3+77*
.000000000000000000004000000-26+1*
.000005440000050050000000000-23+5*
.000400000000000000000000000-16+1*
095995440090050950884919909-3+78*
.000000000000000000004000000-26+1*
.000005440000050050000000000-23+5*
.005000000000000000000000000-13+1*
195995440090450950884919909-3+79*
.000000000000000000004000000-26+1*
.000005440000050050000000000-23+5*
.005000000000400000000000000-15+2*
844000551450108224440801828-4+80*
.000000000400000004000000000-23+2*
.000000000050000000440000000-24+3*
.000000550000000000000000000-14+2*
.044000000000000000000000000-13+2*
988000551490909024880909909-2+81*
.000000000400000004000000000-23+2*
.000000550000000000000000000-14+2*
055180551450002824440801828-4+82*
.000000000400000004000000000-23+2*
.000000000050000000440000000-24+3*
.000000550000000000000000000-14+2*
.055000000000000000000000000-13+2*
199180551450808824440801828-3+83*
.000000000400000004000000000-23+2*
.000000000050000000440000000-24+3*

.0000005500000000000000000000-14+2*
8848105514501088244408018 28-4+84*
.0000000004000000040000000000-23+2*
.0000000000500000004400000000-24+3*
.0000005500000000000000000000-14+2*
.0040000000000000000000000000-13+1*
9884105514909090248809099 09-3+85*
.0000000004000000040000000000-23+2*
.0004000000000000000000000000-16+1*
.0000005500000000000000000000-14+2*
0959905514900009248809099 09-3+86*
.0000000004000000040000000000-23+2*
.0000005500000000000000000000-14+2*
.0050000000000000000000000000-13+1*
1959905514904009248809099 09-3+87*
.0000000004000000040000000000-23+2*
.0050000000004000000000000000-15+2*
.0000005500000000000000000000-14+2*
8440015514501482444408018 28-3+88*
.0000005504000400440000000000-23+6*
.0000000000500000004400000000-24+3*
.0440000000000000000000000000-13+2*
9880015514909490448809099 09-1+89*
.0000005504000400440000000000-23+6*
0551815514500428444408018 28-3+90*
.0000005504000400440000000000-23+6*
.0000000000500000004400000000-24+3*
.0550000000000000000000000000-13+2*
1991815514508488444408018 28-2+91*
.0000005504000400440000000000-23+6*
.0000000000500000004400000000-24+3*
8848115514501488444408018 28-3+92*
.0000005504000400440000000000-23+6*
.0000000000500000004400000000-24+3*
.0040000000000000000000000000-13+1*
9884115514909490448809099 09-2+93*
.0000005504000400440000000000-23+6*
.0004000000000000000000000000-16+1*
0959915514900409448809099 09-2+94*
.0000005504000400440000000000-23+6*
.0050000000000000000000000000-13+1*
1959915514904409448809099 09-2+95*
.0000005504000400440000000000-23+6*
.0050000000004000000000000000-15+2*
8440044441501182444408018 28-4+96*
.0000044040000000440000000000-23+5*
.0000000000500000004400000000-24+3*
.0000004400000000000000000000-14+2*
.0440000000000000000000000000-13+2*
9880044441909190448809099 09-2+97*
.0000044040000000440000000000-23+5*
.0000004400000000000000000000-14+2*
0551844441500128444408018 28-4+98*
.0000044040000000440000000000-23+5*
.0000000000500000004400000000-24+3*
.0000004400000000000000000000-14+2*
.0550000000000000000000000000-13+2*
1991844441508188444408018 28-3+99*
.0000044040000000440000000000-23+5*
.0000000000500000004400000000-24+3*
.0000004400000000000000000000-14+2*
8848144441501188444408018 28-4+100*
.0000044040000000440000000000-23+5*
.0000000000500000004400000000-24+3*
.0000004400000000000000000000-14+2*
.0040000000000000000000000000-13+1*
9884144441909190448809099 09-3+101*
.0000044040000000440000000000-23+5*
.0004000000000000000000000000-16+1*
.0000004400000000000000000000-14+2*
0959944441900109448809099 09-3+102*
.0000044040000000440000000000-23+5*
.0000004400000000000000000000-14+2*
.0050000000000000000000000000-13+1*
1959944441904109448809099 09-3+103*
.0000044040000000440000000000-23+5*
.0050000000004000000000000000-15+2*
.0000004400000000000000000000-14+2*
8440054441901582508488118 88-4+104*
.0000054400000050050000000000-23+5*
.0000000000000000000040000000-24+1*
.0000004040000000000000000000-18+2*
.0440000000000000000000000000-13+2*
9880054441909590508849199 09-3+105*
.0000000000000000000004000000-26+1*
.0000054400000050050000000000-23+5*
.0000004040000000000000000000-18+2*
0551854441900528508488118 88-4+106*
.0000054400000050050000000000-23+5*
.0000000000000000000040000000-24+1*
.0000004040000000000000000000-18+2*
.0550000000000000000000000000-13+2*
1991854441908588508488118 88-3+107*
.0000054400000050050000000000-23+5*
.0000000000000000000040000000-24+1*
.0000004040000000000000000000-18+2*
8848154441901588508488118 88-4+108*
.0000054400000050050000000000-23+5*
.0000000000000000000040000000-24+1*
.0000004040000000000000000000-18+2*
.0040000000000000000000000000-13+1*
9884154441909590508849199 09-4+109*
.0000000000000000000004000000-26+1*
.0000054400000050050000000000-23+5*
.0000004040000000000000000000-18+2*
.0004000000000000000000000000-16+1*
0959954441900509508849199 09-4+110*
.0000000000000000000004000000-26+1*
.0000054400000050050000000000-23+5*
.0000004040000000000000000000-18+2*
.0050000000000000000000000000-13+1*
1959954441904509508849199 09-4+111*
.0000000000000000000004000000-26+1*
.0000054400000050050000000000-23+5*
.0050000000004000000000000000-15+2*
.0000004040000000000000000000-18+2*

844000555590108205848811888-4+112*
.000000505500000005000000000-23+4*
.000000000000000000040000000-24+1*
.000000550000000000000000000-14+2*
.044000000000000000000000000-13+2*
988000555590909005884919909-3+113*
.000000000000000000004000000-26+1*
.000000505500000005000000000-23+4*
.000000550000000000000000000-14+2*
055180555590002805848811888-4+114*
.000000505500000005000000000-23+4*
.000000000000000000040000000-24+1*
.000000550000000000000000000-14+2*
.055000000000000000000000000-13+2*
199180555590808805848811888-3+115*
.000000505500000005000000000-23+4*
.000000000000000000040000000-24+1*
.000000550000000000000000000-14+2*
884810555590108805848811888-4+116*
.000000505500000005000000000-23+4*
.000000000000000000040000000-24+1*
.000000550000000000000000000-14+2*
.004000000000000000000000000-13+1*
988410555590909005884919909-4+117*
.000000000000000000004000000-26+1*
.000000505500000005000000000-23+4*
.000400000000000000000000000-16+1*
.000000550000000000000000000-14+2*
095990555590000905884919909-4+118*
.000000000000000000004000000-26+1*
.000000505500000005000000000-23+4*
.000000550000000000000000000-14+2*
.005000000000000000000000000-13+1*
195990555590400905884919909-4+119*
.000000000000000000004000000-26+1*
.000000505500000005000000000-23+4*
.005000000000400000000000000-15+2*
.000000550000000000000000000-14+2*
844001555590148205848811888-4+120*
.000000505500000005000000000-23+4*
.000000000000000000040000000-24+1*
.000000550000040000000000000-17+3*
.044000000000000000000000000-13+2*
988001555590949005884919909-3+121*
.000000000000000000004000000-26+1*
.000000505500000005000000000-23+4*
.000000550000040000000000000-17+3*
055181555590042805848811888-4+122*
.000000505500000005000000000-23+4*
.000000000000000000040000000-24+1*
.000000550000040000000000000-17+3*
.055000000000000000000000000-13+2*
199181555590848805848811888-3+123*
.000000505500000005000000000-23+4*
.000000000000000000040000000-24+1*
.000000550000040000000000000-17+3*
884811555590148805848811888-4+124*
.000000505500000005000000000-23+4*
.000000000000000000040000000-24+1*
.000000550000040000000000000-17+3*
.004000000000000000000000000-13+1*
988415555909490058849199O9-4+125*
.000000000000000000004000000-26+1*
.000000505500000005000000000-23+4*
.000000550000040000000000000-17+3*
.000400000000000000000000000-16+1*
095991555590040905884919909-4+126*
.000000000000000000004000000-26+1*
.000000505500000005000000000-23+4*
.000000550000040000000000000-17+3*
.005000000000000000000000000-13+1*
195991555590440905884919909-4+127*
.000000000000000000004000000-26+1*
.000000505500000005000000000-23+4*
.000000550000040000000000000-17+3*
.005000000000400000000000000-15+2*
844004440005118242440801828-4+128*
.000004000000000040000000000-23+2*
.000000000005000000440000000-24+3*
.000000440000000000000000000-14+2*
.044000000000000000000000000-13+2*
988004440009919042880909909-2+129*
.000004000000000040000000000-23+2*
.000000440000000000000000000-14+2*
055184440005012842440801828-4+130*
.000004000000000040000000000-23+2*
.000000000005000000440000000-24+3*
.000000440000000000000000000-14+2*
.055000000000000000000000000-13+2*
199184440005818842440801828-3+131*
.000004000000000040000000000-23+2*
.000000000005000000440000000-24+3*
.000000440000000000000000000-14+2*
884814440005118842440801828-4+132*
.000004000000000040000000000-23+2*
.000000000005000000440000000-24+3*
.000000440000000000000000000-14+2*
.004000000000000000000000000-13+1*
988414440009919042880909909-3+133*
.000004000000000040000000000-23+2*
.000400000000000000000000000-16+1*
.000000440000000000000000000-14+2*
095994440009010942880909909-3+134*
.000004000000000040000000000-23+2*
.000000440000000000000000000-14+2*
.005000000000000000000000000-13+1*
195994440009410942880909909-3+135*
.000004000000000040000000000-23+2*
.005000000000400000000000000-15+2*
.000000440000000000000000000-14+2*
844005440009158250848811888-3+136*
.000005440000050050000000000-23+5*
.000000000000000000040000000-24+1*
.044000000000000000000000000-13+2*
988005440009959050884919909-2+137*
.000000000000000000004000000-26+1*

.0000054400000050050000000000-23+5*
055185440009052850848811888-3+138*
.0000054400000050050000000000-23+5*
.0000000000000000000040000000-24+1*
.055000000000000000000000000-13+2*
199185440009858850848811888-2+139*
.0000054400000050050000000000-23+5*
.0000000000000000000040000000-24+1*
884815440009158850848811888-3+140*
.0000054400000050050000000000-23+5*
.0000000000000000000040000000-24+1*
.004000000000000000000000000-13+1*
988415440009959050884919909-3+141*
.0000000000000000000004000000-26+1*
.0000054400000050050000000000-23+5*
.000400000000000000000000000-16+1*
095995440009050950884919909-3+142*
.0000000000000000000004000000-26+1*
.0000054400000050050000000000-23+5*
.005000000000000000000000000-13+1*
195995440009450950884919909-3+143*
.0000000000000000000004000000-26+1*
.0000054400000050050000000000-23+5*
.005000000000400000000000000-15+2*
844000551405108224440801828-4+144*
.0000000004000000040000000000-23+2*
.0000000000050000000440000000-24+3*
.000000550000000000000000000-14+2*
.044000000000000000000000000-13+2*
988000551409909024880909909-2+145*
.0000000004000000040000000000-23+2*
.000000550000000000000000000-14+2*
055180551405002824440801828-4+146*
.0000000004000000040000000000-23+2*
.0000000000050000000440000000-24+3*
.000000550000000000000000000-14+2*
.055000000000000000000000000-13+2*
199180551405808824440801828-3+147*
.0000000004000000040000000000-23+2*
.0000000000050000000440000000-24+3*
.000000550000000000000000000-14+2*
884810551405108824440801828-4+148*
.0000000004000000040000000000-23+2*
.0000000000050000000440000000-24+3*
.000000550000000000000000000-14+2*
.004000000000000000000000000-13+1*
988410551409909024880909909-3+149*
.0000000004000000040000000000-23+2*
.000400000000000000000000000-16+1*
.000000550000000000000000000-14+2*
095990551409000924880909909-3+150*
.0000000004000000040000000000-23+2*
.000000550000000000000000000-14+2*
.005000000000000000000000000-13+1*
195990551409400924880909909-3+151*
.0000000004000000040000000000-23+2*
.005000000000400000000000000-15+2*
.000000550000000000000000000-14+2*

844001551405148244440801828-3+152*
.0000005504000400440000000000-23+6*
.0000000000050000004400000000-24+3*
.044000000000000000000000000-13+2*
988001551409949044880909909-1+153*
.0000005504000400440000000000-23+6*
055181551405042844440801828-3+154*
.0000005504000400440000000000-23+6*
.0000000000050000004400000000-24+3*
.055000000000000000000000000-13+2*
199181551405848844440801828-2+155*
.0000005504000400440000000000-23+6*
.0000000000050000004400000000-24+3*
884811551405148844440801828-3+156*
.0000005504000400440000000000-23+6*
.0000000000050000004400000000-24+3*
.004000000000000000000000000-13+1*
988411551409949044880909909-2+157*
.0000005504000400440000000000-23+6*
.000400000000000000000000000-16+1*
095991551409040944880909909-2+158*
.0000005504000400440000000000-23+6*
.005000000000000000000000000-13+1*
195991551409440944880909909-2+159*
.0000005504000400440000000000-23+6*
.005000000000400000000000000-15+2*
844004444105118244440801828-4+160*
.0000044040000000440000000000-23+5*
.0000000000050000004400000000-24+3*
.000000440000000000000000000-14+2*
.044000000000000000000000000-13+2*
988004444109919044880909909-2+161*
.0000044040000000440000000000-23+5*
.000000440000000000000000000-14+2*
055184444105012844440801828-4+162*
.0000044040000000440000000000-23+5*
.0000000000050000004400000000-24+3*
.000000440000000000000000000-14+2*
.055000000000000000000000000-13+2*
199184444105818844440801828-3+163*
.0000044040000000440000000000-23+5*
.0000000000050000004400000000-24+3*
.000000440000000000000000000-14+2*
884814444105118844440801828-4+164*
.0000044040000000440000000000-23+5*
.0000000000050000004400000000-24+3*
.000000440000000000000000000-14+2*
.004000000000000000000000000-13+1*
988414444109919044880909909-3+165*
.0000044040000000440000000000-23+5*
.000400000000000000000000000-16+1*
.000000440000000000000000000-14+2*
095994444109010944880909909-3+166*
.0000044040000000440000000000-23+5*
.000000440000000000000000000-14+2*
.005000000000000000000000000-13+1*
195994444109410944880909909-3+167*
.0000044040000000440000000000-23+5*

.00500000000040000000000000-15+2*
.00000044000000000000000000-14+2*
84400544410915825084881188-4+168*
.00000544000005005000000000-23+5*
.00000000000000000000400000-24+1*
.00000040400000000000000000-18+2*
.04400000000000000000000000-13+2*
98800544410995905088491990-3+169*
.00000000000000000000400000-26+1*
.00000544000005005000000000-23+5*
.00000040400000000000000000-18+2*
.05500000000000000000000000-13+2*
19918544410985850848811888-3+171*
.00000544000005005000000000-23+5*
.00000000000000000000400000-24+1*
.00000040400000000000000000-18+2*
88481544410915885084881188-4+172*
.00000544000005005000000000-23+5*
.00000000000000000000400000-24+1*
.00000040400000000000000000-18+2*
.00400000000000000000000000-13+1*
98841544410995905088491990-4+173*
.00000000000000000000400000-26+1*
.00000544000005005000000000-23+5*
.00000040400000000000000000-18+2*
.00400000000000000000000000-16+1*
09599544410905095088491990-4+174*
.00000000000000000000400000-26+1*
.00000544000005005000000000-23+5*
.00000040400000000000000000-18+2*
.00500000000000000000000000-13+1*
19599544410945095088491990-4+175*
.00000000000000000000400000-26+1*
.00000544000005005000000000-23+5*
.00500000000040000000000000-15+2*
.00000040400000000000000000-18+2*
84400055550910820584881188-4+176*
.00000050550000000500000000-23+4*
.00000000000000000000400000-24+1*
.00000055000000000000000000-14+2*
.04400000000000000000000000-13+2*
98800055550990900588491990-3+177*
.00000000000000000000400000-26+1*
.00000050550000000500000000-23+4*
.00000055000000000000000000-14+2*
05518055550900280584881188-4+178*
.00000050550000000500000000-23+4*
.00000000000000000000400000-24+1*
.00000055000000000000000000-14+2*
.05500000000000000000000000-13+2*
19918055550980880584881188-3+179*
.00000050550000000500000000-23+4*
.00000000000000000000400000-24+1*
.00000055000000000000000000-14+2*

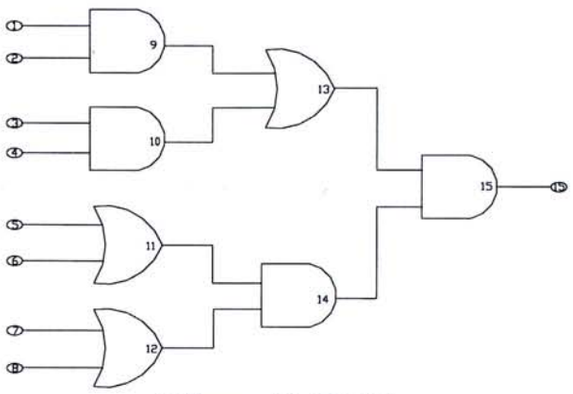
88481055550910880584881188-4+180*
.00000050550000000500000000-23+4*
.00000000000000000000400000-24+1*
.00000055000000000000000000-14+2*
.00400000000000000000000000-13+1*
98841055550990900588491990-4+181*
.00000000000000000000400000-26+1*
.00000050550000000500000000-23+4*
.00040000000000000000000000-16+1*
.00000055000000000000000000-14+2*
09599055550900090588491990-4+182*
.00000000000000000000400000-26+1*
.00000050550000000500000000-23+4*
.00000055000000000000000000-14+2*
.00500000000000000000000000-13+1*
19599055550940090588491990-4+183*
.00000000000000000000400000-26+1*
.00000050550000000500000000-23+4*
.00500000000040000000000000-15+2*
.00000055000000000000000000-14+2*
84400155550914820584881188-4+184*
.00000050550000000500000000-23+4*
.00000000000000000000400000-24+1*
.00000055000040000000000000-17+3*
.04400000000000000000000000-13+2*
98800155550994900588491990-3+185*
.00000000000000000000400000-26+1*
.00000050550000000500000000-23+4*
.00000055000040000000000000-17+3*
05518155550904280584881188-4+186*
.00000050550000000500000000-23+4*
.00000000000000000000400000-24+1*
.00000055000040000000000000-17+3*
.05500000000000000000000000-13+2*
19918155550984880584881188-3+187*
.00000050550000000500000000-23+4*
.00000000000000000000400000-24+1*
.00000055000040000000000000-17+3*
88481155550914880584881188-4+188*
.00000050550000000500000000-23+4*
.00000000000000000000400000-24+1*
.00000055000040000000000000-17+3*
.00400000000000000000000000-13+1*
98841155550994900588491990-4+189*
.00000000000000000000400000-26+1*
.00000050550000000500000000-23+4*
.00000055000040000000000000-17+3*
.00400000000000000000000000-16+1*
09599155550904090588491990-4+190*
.00000000000000000000400000-26+1*
.00000050550000000500000000-23+4*
.00000055000040000000000000-17+3*
.00500000000000000000000000-13+1*
19599155550944090588491990-4+191*
.00000000000000000000400000-26+1*
.00000050550000000500000000-23+4*
.00000055000040000000000000-17+3*
.00500000000040000000000000-15+2*

844004440011118242220801828-3+192*
.000004000000000040000000000-23+2*
.000000440000000000000000000-14+2*
.044000000000000000000000000-13+2*
988004440011919042220903909-2+193*
.000004000000000040000000000-23+2*
.000000440000000000000000000-14+2*
055184440011012842220801828-3+194*
.000004000000000040000000000-23+2*
.000000440000000000000000000-14+2*
.055000000000000000000000000-13+2*
199184440011818842220801828-2+195*
.000004000000000040000000000-23+2*
.000000440000000000000000000-14+2*
884814440011118842220801828-3+196*
.000004000000000040000000000-23+2*
.000000440000000000000000000-14+2*
.004000000000000000000000000-13+1*
988414440011919042220903909-3+197*
.000004000000000040000000000-23+2*
.000400000000000000000000000-16+1*
.000000440000000000000000000-14+2*
095994440011010942220903909-3+198*
.000004000000000040000000000-23+2*
.000000440000000000000000000-14+2*
.005000000000000000000000000-13+1*
195994440011410942220903909-3+199*
.000004000000000040000000000-23+2*
.005000000000400000000000000-15+2*
.000000440000000000000000000-14+2*
844005440011158250222811828-2+200*
.000005440000050050000000000-23+5*
.044000000000000000000000000-13+2*
988005440011959050222913909-1+201*
.000005440000050050000000000-23+5*
055185440011052850222811828-2+202*
.000005440000050050000000000-23+5*
.055000000000000000000000000-13+2*
199185440011858850222811828-1+203*
.000005440000050050000000000-23+5*
884815440011158850222811828-2+204*
.000005440000050050000000000-23+5*
.004000000000000000000000000-13+1*
988415440011959050222913909-2+205*
.000005440000050050000000000-23+5*
.000400000000000000000000000-16+1*
095995440011050950222913909-2+206*
.000005440000050050000000000-23+5*
.005000000000000000000000000-13+1*
195995440011450950222913909-2+207*
.000005440000050050000000000-23+5*
.005000000000400000000000000-15+2*
844000551411108224220801828-3+208*
.000000000400000004000000000-23+2*
.000000550000000000000000000-14+2*
.044000000000000000000000000-13+2*
988000551411909024220903909-2+209*
.000000000400000004000000000-23+2*
.000000550000000000000000000-14+2*
055180551411002824220801828-3+210*
.000000000400000004000000000-23+2*
.000000550000000000000000000-14+2*
.055000000000000000000000000-13+2*
199180551411808824220801828-2+211*
.000000000400000004000000000-23+2*
.000000550000000000000000000-14+2*
884810551411108824220801828-3+212*
.000000000400000004000000000-23+2*
.000000550000000000000000000-14+2*
.004000000000000000000000000-13+1*
988410551411909024220903909-3+213*
.000000000400000004000000000-23+2*
.000400000000000000000000000-16+1*
.000000550000000000000000000-14+2*
095990551411000924220903909-3+214*
.000000000400000004000000000-23+2*
.000000550000000000000000000-14+2*
.005000000000000000000000000-13+1*
195990551411400924220903909-3+215*
.000000000400000004000000000-23+2*
.005000000000400000000000000-15+2*
.000000550000000000000000000-14+2*
844001551411148244220801828-2+216*
.000000550400040044000000000-23+6*
.044000000000000000000000000-13+2*
988001551411949044220903909-1+217*
.000000550400040044000000000-23+6*
055181551411042844220801828-2+218*
.000000550400040044000000000-23+6*
.055000000000000000000000000-13+2*
199181551411848844220801828-1+219*
.000000550400040044000000000-23+6*
884811551411148844220801828-2+220*
.000000550400040044000000000-23+6*
.004000000000000000000000000-13+1*
988411551411949044220903909-2+221*
.000000550400040044000000000-23+6*
.000400000000000000000000000-16+1*
095991551411040944220903909-2+222*
.000000550400040044000000000-23+6*
.005000000000000000000000000-13+1*
195991551411440944220903909-2+223*
.000000550400040044000000000-23+6*
.005000000000400000000000000-15+2*
844004444111118244220801828-3+224*
.000004404000000044000000000-23+5*
.000000440000000000000000000-14+2*
.044000000000000000000000000-13+2*
988004444111919044220903909-2+225*
.000004404000000044000000000-23+5*
.000000440000000000000000000-14+2*
055184444111012844220801828-3+226*
.000004404000000044000000000-23+5*
.000000440000000000000000000-14+2*
.055000000000000000000000000-13+2*
199184444111818844220801828-2+227*

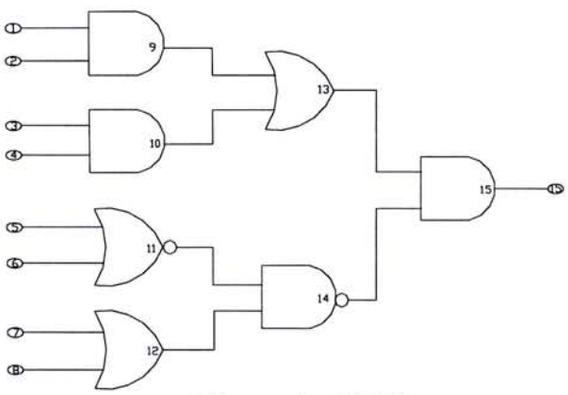
.0000044040000000044000000000-23+5*
.0000004400000000000000000000-14+2*
8848144441111118844220801828-3+228*
.0000044040000000044000000000-23+5*
.0000004400000000000000000000-14+2*
.0040000000000000000000000000-13+1*
9884144441119190442209039O9-3+229*
.0000044040000000044000000000-23+5*
.0040000000000000000000000000-16+1*
.0000004400000000000000000000-14+2*
0959944441110109442209039O9-3+230*
.0000044040000000044000000000-23+5*
.0000004400000000000000000000-14+2*
.0050000000000000000000000000-13+1*
1959944441114109442209039O9-3+231*
.0000044040000000044000000000-23+5*
.0050000000004000000000000000-15+2*
.0000004400000000000000000000-14+2*
8440054441111582502228118 28-3+232*
.0000054400000050050000000000-23+5*
.0000004040000000000000000000-18+2*
.0440000000000000000000000000-13+2*
9880054441119590502229139O9-2+233*
.0000054400000050050000000000-23+5*
.0000004040000000000000000000-18+2*
0551854441110528502228118 28-3+234*
.0000054400000050050000000000-23+5*
.0000004040000000000000000000-18+2*
.0550000000000000000000000000-13+2*
1991854441118588502228118 28-2+235*
.0000054400000050050000000000-23+5*
.0000004040000000000000000000-18+2*
8848154441111588502228118 28-3+236*
.0000054400000050050000000000-23+5*
.0000004040000000000000000000-18+2*
.0040000000000000000000000000-13+1*
9884154441119590502229139O9-3+237*
.0000054400000050050000000000-23+5*
.0000004040000000000000000000-18+2*
.0004000000000000000000000000-16+1*
0959954441110509502229139O9-3+238*
.0000054400000050050000000000-23+5*
.0000004040000000000000000000-18+2*
.0050000000000000000000000000-13+1*
1959954441114509502229139O9-3+239*
.0000054400000050050000000000-23+5*
.0050000000004000000000000000-15+2*
.0000004040000000000000000000-18+2*
8440005555111082052228118 28-3+240*
.0000005055000000050000000000-23+4*
.0000005500000000000000000000-14+2*
.0440000000000000000000000000-13+2*
9880005555119090052229139O9-2+241*
.0000005055000000050000000000-23+4*
.0000005500000000000000000000-14+2*

0551805555110028052228118 28-3+242*
.0000005055000000050000000000-23+4*
.0000005500000000000000000000-14+2*
.0550000000000000000000000000-13+2*
1991805555118088052228118 28-2+243*
.0000005055000000050000000000-23+4*
.0000005500000000000000000000-14+2*
8848105555111088052228118 28-3+244*
.0000005055000000050000000000-23+4*
.0000005500000000000000000000-14+2*
.0040000000000000000000000000-13+1*
9884105555119090052229139O9-3+245*
.0000005055000000050000000000-23+4*
.0004000000000000000000000000-16+1*
.0000005500000000000000000000-14+2*
0959905555110009052229139O9-3+246*
.0000005055000000050000000000-23+4*
.0000005500000000000000000000-14+2*
.0050000000000000000000000000-13+1*
1959905555114009052229139O9-3+247*
.0000005055000000050000000000-23+4*
.0050000000004000000000000000-15+2*
.0000005500000000000000000000-14+2*
8440015555111482052228118 28-3+248*
.0000005055000000050000000000-23+4*
.0000005500000400000000000000-17+3*
.0440000000000000000000000000-13+2*
9880015555119490052229139O9-2+249*
.0000005055000000050000000000-23+4*
.0000005500000400000000000000-17+3*
0551815555110428052228118 28-3+250*
.0000005055000000050000000000-23+4*
.0000005500000400000000000000-17+3*
.0550000000000000000000000000-13+2*
1991815555118488052228118 28-2+251*
.0000005055000000050000000000-23+4*
.0000005500000400000000000000-17+3*
8848115555111488052228118 28-3+252*
.0000005055000000050000000000-23+4*
.0000005500000400000000000000-17+3*
.0040000000000000000000000000-13+1*
9884115555119490052229139O9-3+253*
.0000005055000000050000000000-23+4*
.0000005500000400000000000000-17+3*
.0004000000000000000000000000-16+1*
0959915555110409052229139O9-3+254*
.0000005055000000050000000000-23+4*
.0000005500000400000000000000-17+3*
.0050000000000000000000000000-13+1*
1959915555114409052229139O9-3+255*
.0000005055000000050000000000-23+4*
.0000005500000400000000000000-17+3*
.0050000000004000000000000000-15+2*
End

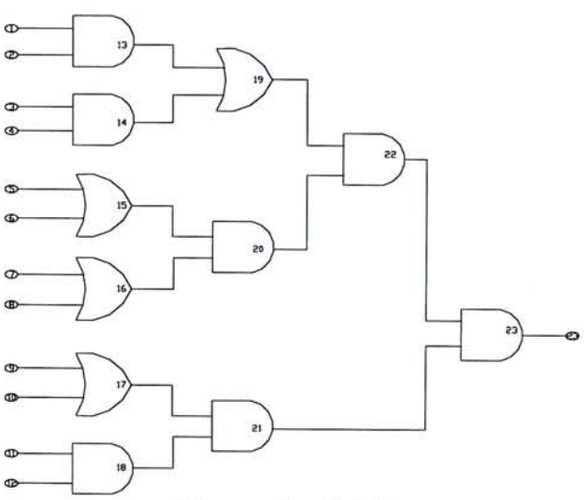
Appendix C



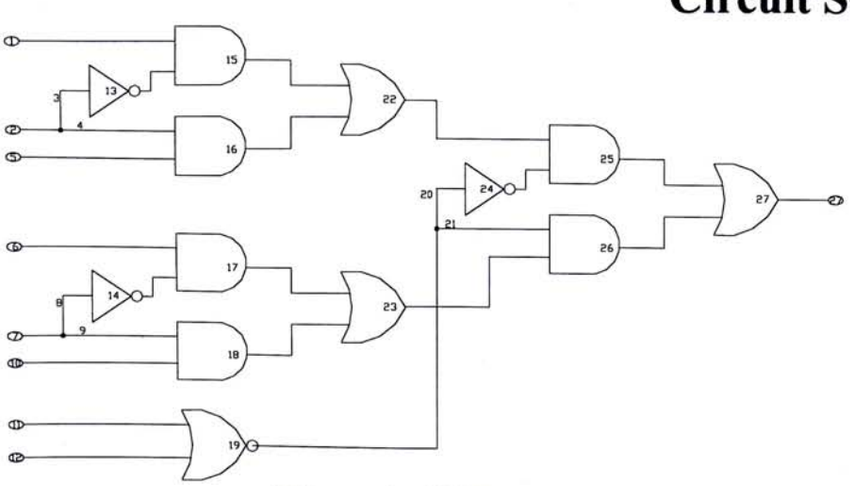
Circuit SC1



Circuit SC2



Circuit SC3



Circuit SC7